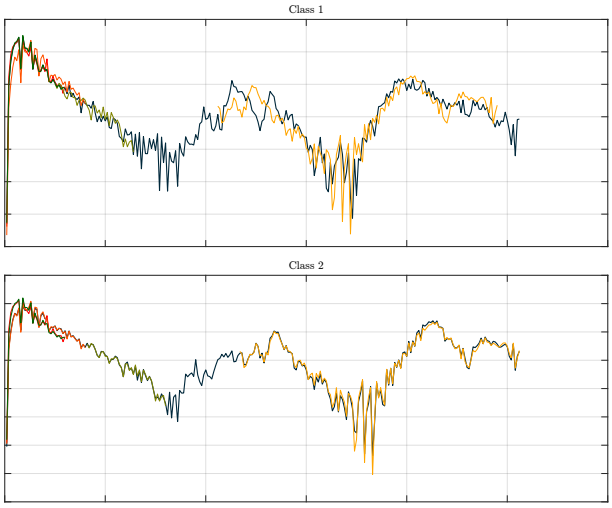
Institut für Mechatronische Systeme
Leibniz Universität Hannover

Leibniz Universität Hannover

Condition Monitoring of Toothed Belt Drives based on Shapelet Classification



Bachelorarbeit B-11/2021-1082

Benjamin Schneg
Matrikelnummer 10013192

Hannover, 18. November 2021

First Examiner	Dr.-Ing. habil. Hans-Georg Jacob
Second Examiner	Prof. Dr.-Ing. Jörg Wallaschek
Supervisor	M.Sc. Moritz Fehsenfeld

Institut für Mechatronische Systeme
Leibniz Universität Hannover

Leibniz Universität Hannover

B-11/2021-1082

Declaration of Authorship

Name:	**Benjamin Schneg**
Registration no.:	10013192
Thesis title:	Condition Monitoring of Toothed Belt Drives based on Shapelet Classification
Type of thesis:	Bachelorarbeit
Study program:	Mechanical Engineering
Submission date:	18. November 2021
First examiner :	Dr.-Ing. habil. Hans-Georg Jacob
Second examiner:	Prof. Dr.-Ing. Jörg Wallaschek
Supervisor:	M.Sc. Moritz Fehsenfeld

I, *Benjamin Schneg*, hereby affirm that the Bachelorarbeit entitled *Condition Monitoring of Toothed Belt Drives based on Shapelet Classification* was written independently, that no references and aids other than those indicated were used, that all passages of the thesis which were taken over literally or analogously from other sources are marked as such and that the thesis has not yet been presented to any examination board in the same or similar form.
I hereby agree to the transmission of my work also to external services for plagiarism checking by plagiarism software.

Hannover, 18. November 2021

(Benjamin Schneg)

Leibniz Universität Hannover

Institut für Mechatronische Systeme
Leibniz Universität Hannover
imes

Bachelor thesis

Mr. Benjamin Schneg, Matr.-No. 10013192

Condition Monitoring of toothed belt drives based on shapelet classification

General description:

The optimum power transmission of a toothed belt drive can only be guaranteed if the toothed belt is optimally pre-tensioned. Incorrect adjustment increases the load on all components involved and reduces efficiency. For this reason, the pre-tensioning force of the belt should be monitored during operation. Condition monitoring of the toothed belt drive is realised via machine learning procedures based on sensor signals from the machine. Feature engineering is an important task during feature-based classification and is crucial for success. Shapelets are subsequences of the original data that are representative for a certain class. Therefore, they can be used for classification and offer the possibility to reveal data insights by highlighting distinguishable parts.

Tasks:

Within the scope of this bachelor thesis, the condition monitoring of toothed belt drives is carried out by means of machine learning based on shapelets. Various reference data sets are available, which can be used to evaluate the performance of all methods. It shall be investigated in which way shapelets can help to classify different belt tensions and gain data insights. For this purpose, suitable methods for the detection of good shapelets will be selected after a literature search. After an investigation of all adjustable parameters of these algorithms, they are evaluated with respect to accuracy and required size of dataset.

In the context of this work, the following tasks arise in particular:

- Literature research on time series classification based on shapelets
- Selection and implementation of a shapelet detection algorithm
- Implementation of a shapelet classification
- Evaluation of all methods with respect to accuracy and data requirement
- Optional: Comparison with handcrafted feature-based approach
- Documentation of the results

The total time for work is 360h.
Date of Issue: 18.05.2021 Submission date: 18.11.2021

Advisor: M. Sc. Moritz Fehsenfeld

....
(Dr.-Ing. Hans-Georg Jacob) (Benjamin Schneg)

Abstract

The need to increase the efficiency of machine operation entails a desire to continuously monitor the condition of crucial machine parts. Therefore, inefficient operation states are detected and improved maintenance procedures are possible.
In this thesis, the core of a condition monitoring system for toothed belt drives is designed. The efficiency of toothed belt drives depends, among other things, on an optimum pre-tension of the belt. For this purpose, a selected machine learning (ML) model is trained to detect different tension states from time series data of the drive provided by internal sensors. To make the data accessible to ML algorithms, a preprocessing that summarizes properties of the data is carried out. Time series shapelets aim to capture local patterns of time series data that characterize the data regarding a desired property. In this thesis, a ML model is trained based on features that are built up on shapelets. Hence, it is investigated to what extent the pre-tension of a belt drive is detectable by small subsequences of operation data such as recordings of the velocity, acceleration, position error and torque. Furthermore, the success of this approach is evaluated regarding the required data size, mere accuracy and potential interpretability enhancements that may be entailed by the shapelet approach.
The results show that satisfactory accuracies are achieved by ML models based on shapelets. For the two data sets that are investigated, accuracies of $ACC_{\mathrm{JLT}} = 84.54\%$ and $ACC_{\mathrm{MFE}} = 99.82\%$ are achieved. Furthermore it is observed that the shapelet discovery process is almost invariant to the size of its used data set. The applied algorithm of shapelet discovery detects even in very small balanced data sets well-performing shapelets. Additionally it is shown that shapelets are a useful tool to visualize parts of time series that are particularly expressive of class membership. Thus, further insights into the problem are possible. Results of comparing ML models trained by shapelets with ML models trained by features that capture global time series properties show that global time series properties seem to be more distinctive.

Contents

List of Figures

List of Tables

Nomenclature

Latin Symbols

ACC	Accuracy
c_m	m-th value of target variable
k	Number of extracted shapelets
l	Length of subsequence
m	Length of time series
m_f	Size of feature subset
n	Size of time series data set
$n_{\min}$	Minimum Node Size
N_{Trees}	Number of trees

Abbreviations

CBM	Condition-based Maintenance
CM	Condition Monitoring
FFT	Fast Fourier Transform
JLT	Jerk-limited trajectory
MFE	Multi Frequency Excitation
ML	Machine Learning
TBM	Time-based Maintenance
TSC	Time Series Classification

1 Introduction

Today's industry is marked by high global competition and an omnipresent desire for growth and innovation. To succeed in this environment, competitors are conducting, among other things, just-in-time production to reduce their inventory [TLB20]. As a consequence, avoidance of production downtimes and machine failures becomes an important task.

Condition monitoring (CM) is one approach to cope with these requirements. It denotes the procedure of processing machine data and gaining insights into the condition of machine components. By that, detecting deviations from the normal operation conditions and diagnosing failures is possible [TLB20]. Important industry 4.0 applications like predictive maintenance rely on functioning CM. Here, maintenance measures highly depend on a machine component's current condition, and in contrast to traditional approaches, they are planned and executed accordingly. When employing predictive maintenance, operation failures become unlikely, a component's lifetime is increased and the continuous operation of machines becomes more efficient. The estimated cost savings resulting from predictive maintenance applications range from 10 to 40 percent [McK15].

Machine learning (ML) algorithms have experienced an upswing in recent years. The general strive for automation, the desire to process available data provided by the progressing digitalization and improvements in the field of microelectronics and hardware are some reasons addressing that trend. The unique selling point of ML algorithms is their ability to self-optimize through data and experience as well as automatic decision-making in a specific domain. Correspondingly, many tasks and challenges are tackled nowadays with means of ML since their principles work on a wide variety of problems. An important subset of ML algorithms is designed to perform classification tasks on input data. Here, a certain category is assigned to input data by the algorithm leading to a gain of information. Coming from real-world applications, the data to be classified is often in chronological order which is referred to as time series data. Feature-based time series classification aims at analyzing such data as it detects patterns and trends that can be used to distinguish and categorize the data. A critical step within this method is successful feature engineering [XPK10]. Features are attributes or variables that can describe data in a summarizing way resulting in a dimension reduction of the data. They can be some scalar values that characterize a time series by assigning specific properties to it. Despite potentially being highly accurate, the interpretability of those kinds of classifiers

is often very low and the classifier's decision-making is not palpable [LDHB12]. To mitigate this problem, [YK11] have introduced a new time series primitive called time series shapelets. Shapelets are subsequences of the original data that are characteristic of a class. Using shapelets as features makes a classifier's decision easier to understand because it enables seeing the actual parts of the data that are critical for a distinction. Thereby new insights into the data may be gained, resulting in a more interpretable classification.

This thesis investigates the usage of shapelets as features for a classifier that carries out CM for toothed belt drives. The classifier is trained with features derived from sensor data such as the velocity, torque, acceleration, etc. recorded by internal sensors inside such a drive. Toothed belt drives are popular machines in the industrial environment and require a specific belt tension to operate efficiently. The classifier is trained to detect the actual tension of the belt and thereby predicts failures or misconditions. It is assessed how the usage of shapelets affects the classifier's accuracy and enables new insights into the data and the general problem. This is done by a comparison between classifiers trained with handcrafted features and classifiers trained with shapelets as features. Furthermore, the data requirement needed to perform accurate classification using this method is investigated. Excluding the introduction and the conclusion, this thesis consists of four chapters. In the first two chapters a theoretic foundation regarding the fields of ML and CM is given. The third chapter outlines the concrete problem tackled in this thesis and describes the implementation of a CM for toothed belt drives. In the fourth chapter, all methods and experiments are evaluated.

2 Fundamentals of Machine Learning and Data-Driven Modeling

Because methods of the presented thesis are based on ML algorithms, a basic understanding of these procedures is provided. In this chapter, a general introduction to ML methods is given and the different approaches within the ML domain are described and explained. Additionally, models and procedures explicitly applied in this thesis are discussed in more detail.

2.1 Introduction to Machine Learning Algorithms

Conventional computer programs are rigid and usually designed to perform one specific task. They mainly consist of detailed instructions on how to solve a certain problem step by step from input to output. Solving the specific problem, they may perform very efficiently. However, conventional programs are constrained to only that problem they were initially designed for. The coordinated interaction between program and input data is an important prerequisite for a functioning program. If one of the two changes, program errors become inevitable. Consequently, system changes need to be done every time the problem domain changes slightly. To give an example, an image processing program that aims for distinguishing humans and animals can be instructed to recognize cats as animals and young men as a human. However, once a dog is presented to the program, it fails because no instructions were made to recognize dogs as animals. In summary, they are unable to generalize.
ML algorithms instead follow a different approach. When tackling a problem, not the input data and a customized program are used to solve that problem. Rather, input data and, in some cases, its desired output are given to the algorithm, which provides then, under some given constraints, a program that is able to deal with the given problem. Figure 2.1 illustrates this difference.
The core idea is to let the algorithm itself detect underlying rules and laws of the problem rather than giving detailed instructions to the computer. By that, a problem can be tackled in a much wider way since these types of algorithms are flexible and learn to adapt under new circumstances. These properties allow denoting such methods as being able to learn. Commonly, ML algorithms are initialized by going through a training phase. The algorithm sees a number of data samples, and thereby an optimization process is being carried out. This process controls the adjustment of the algorithm's parameters and fits the chosen internal model

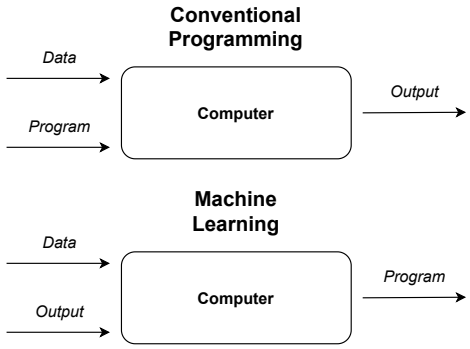


Figure 2.1: Conventional programming paradigm and machine learning [GGB16]

of the algorithm to the training data. After that, the model is ready to process new data instances and is able to cope with the given task. The statement of being able to "learn from experience" refers to the property that ML algorithms commonly improve their accuracy depending on how much data they have been fed during the training phase. A popular way to distinguish ML algorithms is to categorize them according to their learning strategy. Three common categories occur in the literature when classifying ML algorithms:

1. Supervised Learning.
2. Unsupervised Learning.
3. Reinforcement Learning.

All three variations have their own field of use where they work best or even are the only possibility that works. A decisive difference among them is the type of data presented to them during the training phase. Especially supervised learning and unsupervised learning can be distinguished by this. Supervised learners are always trained with labeled data. That is, to every data sample of the training data set, a desired target value is attached. The supervised learning algorithm then fits a model to the training data that maps input variables to the target variable and aims for predicting the target value of unseen data.

Unsupervised learners, on the other hand, only deal with raw unlabeled data. Unlike supervised learners, they do not aim for prediction. Rather, their goal is to discover unknown patterns, tendencies, and relationships between variables so that data samples can be segmented and distinguished. They are used as a preprocess to supervised learning or to carry out a dimension reduction of the data enabling, e.g., visualization of the data [Bis06].

The characterization based on the type of training data does not apply to reinforcement learning. The interpretation of learning applied in this method is the most intuitive among the three variations. In reinforcement learning, the term learning denotes a process of trial and error [PLM96]. Here, a system takes arbitrary actions and compares the outcome of its actions to a given objective. If the outcome brings the system closer to its objective, it is rewarded, and the actions taken are stored. Consequently, wrong actions are not stored, and the reward is marginal. Unlike unsupervised and supervised learning, reinforcement learning does not aim for pattern recognition or prediction. Rather, such systems learn to behave optimally in a certain environment.

In this thesis, only means of supervised learning are applied. Therefore this approach is explained in the following section in more detail. Furthermore, an insight into the chronological sequence of supervised learning methods is given.

2.2 Supervised Learning

The common goal of supervised learning methods is the prediction of a quantitative or qualitative value based on a given input of data samples. For the sake of uniform terminology in this thesis, the input data samples are referred to as *predictors* whereas the respective output is denoted as *target variable*. The relationship between predictors and target variable in the scenario of supervised learning is initially unknown. The following methods and means form the core of supervised learning:

- A labeled data set $\boldsymbol{T} = (\boldsymbol{X}, \boldsymbol{y})$ containing the predictors $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ with m predictors consisting of n elements and a set $\boldsymbol{y} \in \mathbb{R}^{m}$ containing the respective target values of each predictor
- A mathematical model that attempts to approximate the relationship between predictors $\boldsymbol{X}$ and target variable $\boldsymbol{y}$ subject to an error $\boldsymbol{\epsilon}$:

$$\boldsymbol{y} = \hat{f}(\boldsymbol{X}) + \boldsymbol{\epsilon} \tag{2.1}$$

- an optimization process that controls the adjustment of the model's parameters so that it approximates the relationship between predictors and target variable as accurate as possible

Based on this scenario, the supervised learning process is usually as follows: First, the available data set is divided into a training data set and a test data set. Subsequently, a preprocessing

of the data is performed. In some cases, a dimension reduction of the data is needed, or a transformation into more meaningful domains is required for successful model fitting. This procedure of preprocessing is referred to as *feature engineering* and is further explained in section 2.2.2. After the feature engineering, the actual process of model fitting is carried out. Therefore a suitable model for the application is to be selected, and the training phase is initialized. During the training phase, an optimization process adapted to the model takes place. From every training data sample, the model can derive information about the relationship between predictors and target variable. This is done by adjusting parameters of the model so that it depicts the training data as accurately as possible. Intuitively, the more data samples are presented during the training phase the more accurate the model maps the relationship between predictors and target variable. In the domain of supervised learning, a distinction between models based on the type of target variable they aim to predict is common: *Classification* deals with data whose target variable is of qualitative nature. Here, the goal is to assign a specific category to the predictors. In contrast, models that aim for prediction of quantitative target variables are assigned to the area of *regression*. Since the target variable in this thesis is categorical, in section 2.2.3 further information about *classification models* and those that are explicitly used in this thesis is provided. Depending on the selected model's properties often some parameters independent from the application also need to be tuned. Section 2.2.4 introduces methods that aim to tune those parameters, which are also referred to as *hyperparameters*. Since only the training data set is involved in the training process, the model's accuracy can be assessed with the test data set. Data samples from this set are inserted into the generated model, which then predicts target values for every test data sample. The model's accuracy can be derived by comparing the predicted values to the real values. This process of training and testing the model is carried out several times to rule out statistic uncertainties and obtain a robust result. In section 2.2.1 a further explanation of this procedure is provided, which is also referred to as *resampling*. Additionally, means to describe a model's accuracy itself are presented in section 2.2.5.

When choosing a model, an important and influential property for the quality of prediction is the model's complexity. It is crucial for the ability of generalization and needs to be set carefully [HTF09]. The complexity, also known as flexibility, can be described, for instance, by the number of parameters to be adjusted or the degree of freedom the selected model offers. In general, the higher the model's flexibility, the more it adapts to the training data. Contrary to a first assumption, high flexibility does not always correspond to an accurate model. If a model's flexibility is too high, it may adapt to local deviations of the training data since data is always affected by noise. That is, it reproduces properties from the training data very precisely, including noisy and inaccurate samples of predictors and their respective target values. In that

case, the model fails to learn the general relationship between predictors and target variables. This scenario is also known as *overfitting* [JWHT13]. To generate an accurate model, the extent of fit to the training data, which is influenced by the model's flexibility, is to be adjusted carefully. Two quantities denoted as *bias* and *variance* are of importance here. Bias refers to the training error and thus to the extent of fit. Variance is proportional to flexibility. Models with a high variance tend to change rapidly when being fitted to different data sets [JWHT13]. Low-variance-models instead commonly maintain a general form and change rather slightly when fitted to different data sets. A model with low bias and low variance would be beneficial, but since these two quantities contrarily influence each other, a compromise must be found. This compromise is known as *bias-variance trade-off* and is illustrated in Figure 2.2. It is shown that optimal flexibility is to be found to maximize a model's accuracy.

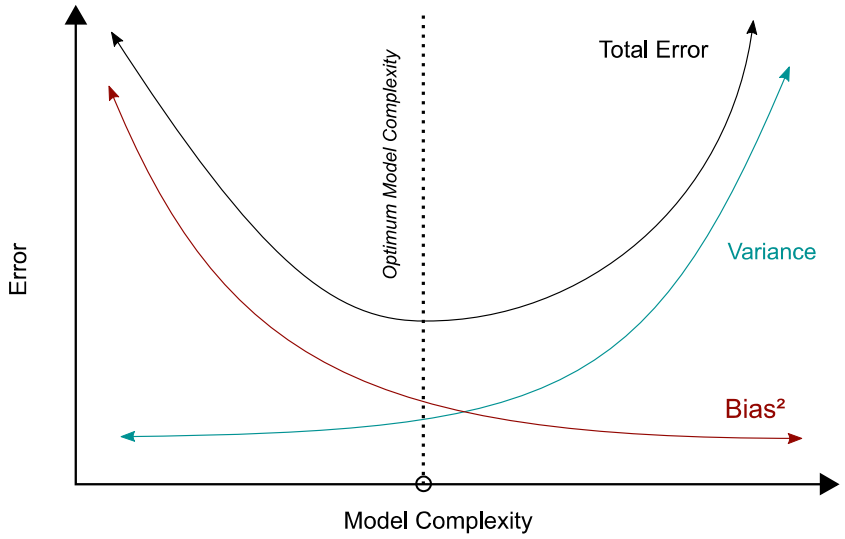


Figure 2.2: Bias-Variance Trade-Off [FR12]

To enable an optimal selection of a model, it is crucial to conduct a robust evaluation of the model's accuracy. Based on that evaluation, a decision can be derived if the selected model is suitable for the application or not. Resampling methods aim to provide robust evaluation methods and are presented in the following section.

2.2.1 Resampling

Before training a model, the available data first needs to be divided into test and training data sets. As already mentioned, a high amount of training data increases the probability of obtaining a well predicting model. On the other hand, a robust evaluation of the model's accuracy also

relies on a sufficient amount of data. This problem of reliably estimating a model's accuracy while providing enough data for successful training is tackled with methods of resampling.

The idea of *cross-validation* is to split the data several times into test data and training data. For every split, a model is trained and subsequently evaluated by the test error. *K-Fold Cross-Validation* proposes to split the data into k equal-sized parts. After that, a model is trained k-times using $k-1$ portions of the data for training and one portion as test data. As illustrated in Figure 2.3, for each run, a different portion of the data is used as test data [JWHT13]. A robust model evaluation is obtained by averaging the resulting test errors. Considering a *k-fold cross-validation* generating k test errors ERR_i, the final error estimation is given by

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} ERR_i. \tag{2.2}$$

This approach enables using a large proportion of the data as training data while still being able to reliably estimate a model's accuracy. [Koh95] suggests to split the data in 10 portions for the most reliable and efficient error estimation.

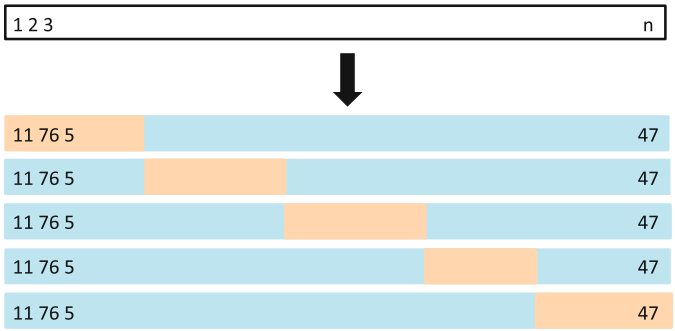


Figure 2.3: 5-Fold Cross Validation [JWHT13]

[Efr79] introduced another important resampling method, called the *bootstrap*. Here, equal-sized subsets of the available data $\boldsymbol{Z}$, the bootstrap data sets $\{\boldsymbol{Z}^{*1}, \boldsymbol{Z}^{*2}, ..., \boldsymbol{Z}^{*B}\}$, are generated by sampling the data B-times [HTF09]. The sampling is carried out *with replacement*. Hence, a single data point can occur several times in a bootstrap data set. The bootstrap data sets maintain the statistical properties of the original distribution on average and can be used for robust error estimation. In Figure 2.4 a rough overview of bootstrapping is displayed. Samples of the original data set are created. After that, a certain statistical quantity S can be robustly computed by drawing this quantity from every bootstrap sample $\{S(\boldsymbol{Z}^{*1}), S(\boldsymbol{Z}^{*2}), ..., S(\boldsymbol{Z}^{*B})\}$

and averaging the results subsequently. Insights, how this method is used concretely for building predictive models are provided in section 2.2.3.

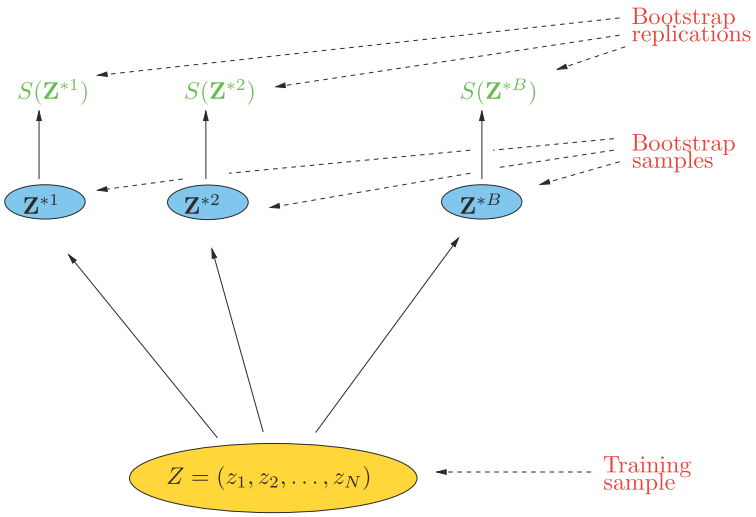


Figure 2.4: Schematic of the bootstrap process [HTF09]

2.2.2 Feature Engineering

When training predictive models with data, often the data needs to be preprocessed for a successful and efficient model generation. That is, distinctive information is not always directly available in the data but must be extracted from the data. Additionally, specific models and procedures require the data in a certain form or dimension [OK20]. *Features* are variables that are distinctive for the target variable and, if required, tuned to a specific model. Their behavior correlates to the target variable, and thus, they are useful for training predictive models. The procedure of *feature engineering* consists of creating features, transforming the available data into the feature space, and selecting features for training a model. Any quantity derived from the original data may be useful as a feature. What quantity may be considered a feature and what it consists of is either decided manually or even automatically by the algorithm. The manual creation and selection of features offer the possibility to contribute domain knowledge [OK20]. Findings of relationships between predictor and target variable that are already available can be integrated by drawing features from the original data that effectively capture this knowledge. The process of feature engineering is a crucial step for a successful application of ML algorithms because it makes the data accessible for learning algorithms. Loosely speaking, feature engineering filters or enhances the data so that it only

consists of information that are maximally distinctive for the target variable. Another benefit from features is their contribution to a higher interpretability of the generated model. Since features are predefined variables, a model's result can be traced based on each feature's value. If no domain knowledge is available, the influence of features on the target variable can be assessed by statistical tests (e.g. *t-statistic* or *F-statistic*) after training a model [JWHT13]. In this way, features can be selected despite the lack of domain knowledge.

2.2.3 Supervised Classification Models

In this section, two learning algorithms that build on each other are examined and presented. *Decision trees* and *random forests* are two well-known algorithms that are particularly common for classification tasks. The random forest, which is used in this thesis, is an enhancement of decision trees and is presented after a discussion of decision trees.

Decision trees build a predictive model by partitioning the feature space into cuboid regions that correspond to a certain target variable [Bis06]. The assignment of predictors to a partition and thus to a class is based on the approval or refusal of a fixed sequence of logical conditions. In order to achieve a basic understanding of decision trees, it is helpful first to take a look at the structure. Different decision tree topologies exist, however, the following explanations regarding decision trees refer to *binary* decision trees. As illustrated in Figure 2.5, decision trees consist of a structured arrangement of *nodes* that branch usually into two further nodes. This structure evolves until nodes do not branch any further and end up in a *leaf*. Nodes that are connected to other nodes are called *parent node* or *child node* depending on the perspective. Decision trees are structured according to the top-down principle. They start with a root node and branch until the leaves. Every node contains a logical statement that is either true or false depending on what it is exposed to. Classification via decision trees functions as follows: A predictor is classified by the path it takes from root to leaf along the decision tree. The path is determined by the results the predictor produces at every node after the exposure to the respective logical statement. Depending on a match or mismatch to the logical statement, the predictor's further direction is set. If it meets the condition, it would be, e.g., exposed to the right child node. If not, the left child node would be its next destination. This process continues until the predictor reaches a node that is not connected to another. This last node, a leaf, determines the class of the predictor. During training, nodes of decision trees are constructed by finding the best split S in the data that best minimizes the impurity of the resulting two datasets. That is, the data portions are divided according to their class membership. The goal for every split is to enrich the resulting data portions with data of the same class so that in leaf nodes, data of the same class are highly concentrated. This best split is achieved by the logical statement that best divides the data into its classes. Given a dataset $\boldsymbol{T} \in \mathbb{R}^{m \times n}$ containing

$j = m$ predictors of $i = n$ features, logical conditions are built by a pair of one feature f_i and a defined threshold $f_i \leq \theta_0$ to this feature. The key to the optimal split is to find the optimal pair of feature and threshold. This is achieved by maximizing the *information gain*

$$IG = H(\boldsymbol{D}) - \hat{H}(\boldsymbol{D}) \tag{2.3}$$

obtained by a performed split [YK11]. It is computed by comparing the *entropy* H of a dataset $\boldsymbol{D}$ before the split with the resulting entropy $\hat{H}$ after the split. Considering a n-class scenario with classes $K_1, K_2, ..., K_n$ and their respective fractions $p(K_1), p(K_2), ..., p(K_n)$ in the data, the entropy

$$H(\boldsymbol{D}) = -\sum_{i=1}^{n} p(K_i) \log(p(K_i)) \tag{2.4}$$

is a measure of impurity in a data set [YK11]. The term maximizes when the fraction of each class is of equal size. The entropy of the two remaining data sets $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ is obtained by

$$\hat{H}(\boldsymbol{D}) = f(\boldsymbol{D}_1)H(\boldsymbol{D}_1) + f(\boldsymbol{D}_2)H(\boldsymbol{D}_2) \tag{2.5}$$

with $f(\boldsymbol{D}_1)$ and $f(\boldsymbol{D}_2)$ being the respective fraction of the original data set [YK11]. Besides the entropy, the *gini-index* is another common measure to quantify the impurity [Bis06]. Despite its importance in decision tree algorithms, no further elaboration in this regard is made since this section only aims to provide basic knowledge. Decision trees are grown in a greedy

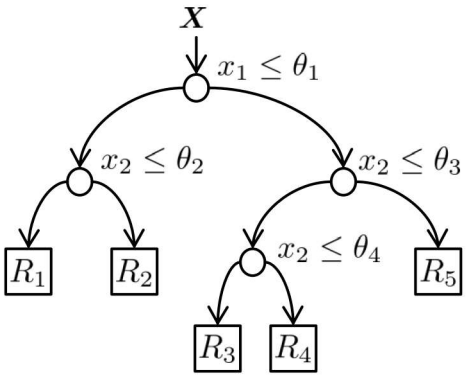


Figure 2.5: Structure of decision trees [Feh19]

fashion, i.g. for every node, the best found split is performed. After a split is set, it remains and is not modified further. Continuing to split the data in this manner, the homogeneity of the resulting data portions after each node increases. Simultaneously, the number of predictors per data portion decreases. During the growing phase, a criterion needs to be applied when to

stop the growing process. A common practice is to set a threshold based on the number of data points associated with the leaf nodes [Bis06]. It is denoted as the minimum node size $n_{\min}$. Given this parameter for training a decision tree, a split is only performed if its remaining data portions do not fall below the set threshold. Once this threshold is reached at every node in the tree, the growing process stops. In general, decision trees are models that are burdened with high variance [JWHT13]. The deeper a decision tree is grown, i.g. the more nodes it contains, the more it tends to overfit. Since every further split is performed to a smaller portion of data during the training phase, the assessment, if that split is in fact a general class distinctive rule, is subject to increasing uncertainty. *Pruning* the decision tree aims at reducing its variance with the undesired side effect of a bias increase. However, these properties make the decision tree a weak learner.

Nevertheless, research was done to mitigate the problem of decision trees being affected by a strong bias-variance trade-off. *Bagging*, introduced by [Bre96], is one approach that successfully alleviates the variance while maintaining a low bias. It stems from the idea of applying the *bootstrap* to statistical learners like decision trees. From the original training data set, bootstrap samples are drawn and used to train decision trees in a deep fashion, i.g. with low bias. While one single deeply grown decision tree is subject to high variance and thus inaccurate, an aggregation of those trees combined to one single model turns out to decrease the variance significantly [HTF09]. Classification with bagged trees is conducted by a *majority vote*. Every tree of the ensemble votes for a class, the class with the most votes wins and is assigned to the predictor. The term ***bagging*** is an acronym of the expression "**b**ootstrap **agg**regat**ing**" which summarizes the core idea of this method [Bre96]. Given a set of $b = 1, 2, ..., B$ bootstrap samples $\boldsymbol{Z}^{*b}$ and let $\hat{C}(x)^{*b}$ be the class prediction of the b-th decision tree of the aggregation, then, according to [HTF09], the bagging estimate is obtained by

$$\hat{C}_{\text{bag}}(x)^{*b} = \text{majority vote } \{\hat{C}(x)^{*b}\}_1^B. \tag{2.6}$$

In bagging, each decision tree is grown traditionally, i.g. always the most distinctive pairs of feature and threshold are selected for the first splits. Since the strongest features are often a small fraction of the whole training data set, the probability that those features remain in many bootstrap data sets is high. As a consequence, the grown decision trees of the bootstrapped aggregation are very similar and suffer from high correlation. This circumstance diminishes the accuracy-boosting variance reduction of the bagging method. In order to further reduce variance and thus increase accuracy *random forests*, introduced by [Bre01], involve aggregating *de-correlated* trees and perform a majority vote with them (see Figure 2.6). Growing de-correlated trees is accomplished by restricting the feature set that is available for every split

during the growing process. For every split during training, a feature subset is drawn at random. The size m_f of this set is to be selected before training and controls, to what extent the decorrelation of trees is conducted. Decreasing m_f results in smaller feature subsets for every split and thus reduces the probability of obtaining similar trees. Considering a bootstrap data set containing p features, the default value for m_f is $\sqrt{p}$ [HTF09]. Since random forests are an aggregation of decision trees, another influential parameter is the minimum node size $n_{\min}$, which controls the depth of each tree and thus its extent of fit to the respective bootstrap set. For classification the default value for $n_{\min}$ is 1 [HTF09]. However, the performance of ML algorithms is highly dependent on the data that they are applied to. A valid, general statement that the suggested default values for these so-called *hyper-parameters* of random forest or other ML algorithms always yield the most accurate models is infeasible. Rather, hyper-parameters must be case-dependent tuned in order to obtain the most precise model.

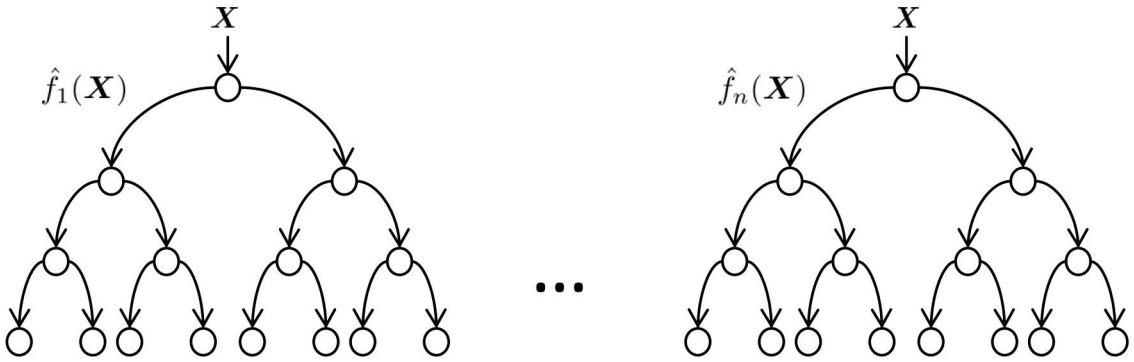


Figure 2.6: Illustration of random forests [Feh19]

2.2.4 Hyperparameter Optimization

In this section, an overview of the field of *hyper-parameter optimization* is provided. Several approaches are introduced, and the optimization method applied in this thesis is outlined in more detail.

The general problem of hyperparameter optimization is as follows: Consider a learning algorithm $\mathcal{A}$ that attempts to find a function f approximating the relationship between predictors and target variable, exemplified by a training data set $\boldsymbol{X}^{(\mathrm{tr})}$, by minimizing an expected loss $\mathcal{L}\big(\boldsymbol{X}^{(\mathrm{te})}; \mathcal{A}(\boldsymbol{X}^{(\mathrm{tr})})\big)$ given that $f = \mathcal{A}(\boldsymbol{X}^{(\mathrm{tr})})$. Now consider that there is not only one single learning algorithm $\mathcal{A}$ of that type but a whole set of these learning algorithms $\boldsymbol{\mathcal{A}}$ that each behave differently due to some prior set hyper-parameters $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ when being exposed to $\boldsymbol{X}^{(\mathrm{tr})}$. Let $\mathcal{A}_{\boldsymbol{\lambda}}$ be the learning algorithm that is to be configured with a set $\boldsymbol{\lambda}$ and let $\mathbb{E}_{\boldsymbol{X}^{(\mathrm{te})}}$ be the

general error of that algorithm. Then the problem of hyper-parameter optimization is to find a hyper-parameter configuration

$$\boldsymbol{\lambda}^{(*)} = \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\operatorname{argmin}} \, \mathbb{E}_{\mathbf{X}^{(\mathrm{te})}}[\mathcal{L}\big(\boldsymbol{X}^{(\mathrm{te})}; \mathcal{A}_{\boldsymbol{\lambda}}(\boldsymbol{X}^{(\mathrm{tr})})\big)] \tag{2.7}$$

that minimizes the general error [BB12]. Recall that very little is known about the general error $\mathbb{E}$ since it is dependent on unforeseeable influences. Different learning algorithms inserted to the error function, varying data applied to a learning algorithm and random noise that occurs due to statistical uncertainties, to name some reasons, prohibit to derive generally valid conclusions about the error function. Therefore an approach to find an automatic hyperparameter optimization is hard to develop since querying the error function at a point $\boldsymbol{\lambda}$ is the only way to evaluate it [BCF]. In the literature, a common approach is indeed a manual search for optimal hyper-parameters, which involves predefined trials. That is, hyper-parameter configurations are previously selected and inserted into the algorithm. Subsequently, every trial is evaluated, and the best one is picked as the optimal configuration. Among manual search strategies, a distinction between grid search and random search is made. According to [BB12], random search outperforms grid search on average. Yet, both strategies are time-consuming and computational intensive because every trial involves training and evaluating a model several times. Nevertheless, the most promising and most applied approach among the ML community is, however, an automatic search. *Sequential Model-Based Bayesian Optimization* (SMBO) has emerged as state-of-the-art hyper-parameter search in some fields of ML as stated by [FSH15]. It connects efficiency and performance since it detects better hyper-parameter configurations using fewer trials than manual search. The core idea of SMBO is to use the information of previously conducted queries on the error function in order to predict regions in the function which are likely to yield low error rates. That is achieved by fitting a probabilistic model to the previously queried values that allows to predict promising trial points. Essentially, SMBO consists of iteratively updating and evaluating the following two functions:

- a *surrogate function* (also called *response surface*) $\mathcal{M}$ that models the error function based on previously queried trials
- an *acquisition function* u that models the utility of every possible hyper-parameter combination $\boldsymbol{\lambda}$ based on the information provided by the model $\mathcal{M}$

The probabilistic model $\mathcal{M}$ does not only predict one certain expected value of the error function based on the conducted queries. Rather, it displays a probabilistic distribution of error values for every possible hyper-parameter combination $\boldsymbol{\lambda}$ by putting out mean μ and variance σ^2 of the expected error distributions. By that the trade-off between *exploration* (global search) and *exploitation* (local search) can be balanced effectively [BCF]. According to the model $\mathcal{M}$,

two regions are interesting for the next query on the error function: Regions that are next to already queried values yielding low error rates and regions subject to high variance σ^2 that may be worthwhile to examine due to here occurring large uncertainty. According to [FSH15], the default acquisition function in SMBO is the *expected positive improvement* (EI)

$$u_{EI}(\boldsymbol{\lambda}, \mathcal{M}) = \int_{-\infty}^{\infty} \max(y^* - y, 0) p_{\mathcal{M}}(y|\boldsymbol{\lambda}) \mathrm{d}y. \tag{2.8}$$

Here, y^* is the error function's best value found so far. The term $p_{\mathcal{M}}(y|\boldsymbol{\lambda})$ denotes the conditional probability that a query on the error function at position $\boldsymbol{\lambda}$ yields the value y. The function measures the utility of all possible hyper-parameter combinations considering both exploitation and exploration. The best compromise of the two yields the maximal utility value of $u_{EI}(\boldsymbol{\lambda}, \mathcal{M})$ and is selected to be the next trial. In Figure 2.7 an exemplary proceeding of a one-dimensional SMBO is illustrated. The dotted line represents the real error function or also called objective function, whereas the continuous black line displays the mean μ of the probabilistic model $\mathcal{M}$. The blue shaded areas around μ illustrate the model's uncertainty corresponding to the variance σ^2. The acquisition function is plotted by the green graph below the objective function. As displayed, the maximum of the acquisition function determines the next trial point on the objective function.

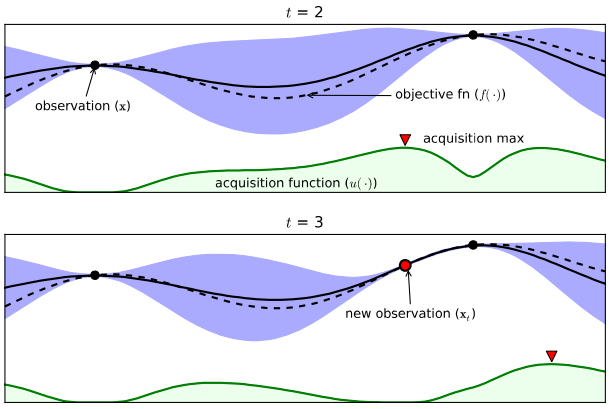


Figure 2.7: Exemplary illustration of a one-dimensional SMBO [BCF]

2.2.5 Evaluation Process

Finally, to complete the discussion on ML processes, a brief instruction on how to evaluate a model's performance is given in this section. Additionally, methods to visualize a learning algorithm's accuracy are presented.

Considering classification problems, a simple and intuitive approach to measure a model's performance is to count how many misclassifications it has brought up after the exposure to a test data set. One obtains the test error rate by putting that number in relation to the absolute number of conducted tests. Given a test data set containing $i = n$ predictors each labeled with y_i, the *test error rate* is obtained by

$$ERR = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{2.9}$$

while the *indicator variable* is defined as

$$I(y_i \neq \hat{y}_i) = \begin{cases} 1, & \text{if } y_i \neq \hat{y}_i \\ 0, & \text{if } y_i = \hat{y}_i \end{cases} \tag{2.10}$$

with $\hat{y}_i$ being the i-th predicted target value of the evaluated model [JWHT13]. An important tool to visualize the results of testing a classifier is the *confusion matrix*. It is often applied to provide a more detailed look at the performance of a classifier. Considering a k-class problem with classes $K_1, K_2, ..., K_k$, the confusion matrix consists of $k \times k$ fields. Let j be the index for the rows and i be the index for the columns in that matrix. Then the field in row $j = m$ and column $i = n$ displays the relative proportion of predictors from class K_m that were misclassified as K_n. Thus, the fields on the diagonal line display the proportions of correctly classified predictors, while the remaining fields display the proportions of misclassified predictors. In contrast to the general error rate, the confusion matrix allows judging a classifier's performance dependent on the individual classes. Statements on which class it performs strongly and on which poorly become possible. In Figure 2.8 an exemplary confusion matrix with 5 classes is displayed.

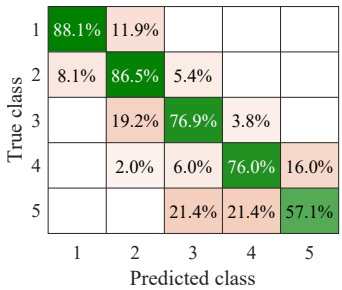


Figure 2.8: Exemplary confusion matrix of a 5-class problem

3 Condition Monitoring and Time Series Classification

This chapter introduces basic concepts of *condition monitoring* (CM), the related field of *time series classification* (TSC) and aims to provide a brief presentation on conducted research in the regarded fields. The research question to be addressed in this thesis will be embedded in the context of condition monitoring. Additionally, terms and methodologies that are necessary to carry out CM as requested for this work are introduced and outlined in more detail. Eventually, in this chapter, the concept of *time series shapelets* is presented and discussed.

3.1 Introduction to Condition Monitoring

Industrial processes and industrial production, in general, are based on functioning machines. A warranty that these machines function during a desired time would ensure both economic efficiency and a safe industrial environment [ZYW19]. Unexpected machine failures entail extensive financial damage and pose a danger for industrial workers. Since competitors in the industrial sector nowadays only sustain with a respective efficiency, cutting costs and efficient production are important aspects for successful operation in the market [AN19]. CM aims to continuously record the health state of critical parts within a machine or a machine itself. Therefore, a machine's operation can be optimized, or decisions regarding maintenance are facilitated. According to [HZTM09] and [MTMZ12] the following approaches of CM exist:

- *Model-based approach*
- *Data-driven approach*

The model-based approach is built up by manually created mathematical models of a system's behavior incorporating physical laws expressed in differential equations. Therefore, they require an expert's domain knowledge to be created. However, since the installation of model-based approaches typically requires an interruption of the machine's operation and is infeasible for large-scale complex systems, they are considered unfavorable in the literature [HCFC$^+$14]. Data-driven approaches instead monitor the condition of machines using models created by means of ML. They can be obtained and handled without an expert's domain knowledge and even cope with large-scale complex systems [ZYW19]. Data-driven approaches utilize recorded data of different operation states to derive a model that is able to depict a machine's condition and thus do not require a previously conducted compressive study of a machine's behavior.

Especially in the field of machine maintenance, CM applications are increasingly established. According to [WCW07], two main approaches within the field of maintenance exist:

- *corrective maintenance*
- *preventive maintenance*

Corrective maintenance is the most basic and oldest approach. Here, machines are operated until failure and get repaired afterward. Obviously, this approach is, despite its simplicity, the most unfavorable due to the large resulting costs of machine failures. Preventive maintenance instead is applied to prevent machine failures using either *time-based maintenance* (TBM) or *condition-based maintenance* (CBM) [AN19]. While TBM involves a predetermined calendar schedule to perform maintenance regardless of a machine's real condition, CBM measures highly depend on a machine's current condition. Thereby, warranties to prevent unexpected machine failures are possible, which is not the case for TBM installations. Obviously, further cost reductions can be achieved with CBM as shown by numerous economic studies [MA07, VKS13]. Furthermore, examinations on the nature of machine failures showed that a majority of equipment failures are preceded by certain conditions indicating that a failure is going to happen [AN19]. Hence, CM systems may even predict failures that will happen in the future and may conduct early detection of failures in order to provide sufficient time to plan and perform necessary maintenance. Another benefit of an accurate CM system is the higher operational efficiency of a machine enabled by CM. Inefficient operation states that occur during long-term operation may be detected and reported to adjust the machine towards a more efficient operation.

3.2 Data-Driven Condition Monitoring

In data-driven CM, models that depict the condition of a machine are built up by means of ML. The collection of data mined from internal sensors of the machine is used to derive inference about the relationship between those parameters and potential failure conditions. Even the condition of large-scale complex systems is possible to monitor using this method [YLGK15]. Data-driven CM is the core of a CBM program as it enables intelligent maintenance decision-making. The operation of machines incorporating a data-driven CM system is as follows: Sensors that are attached to the machine record different operational parameters. This data is then inserted into the created data-driven model. The model is trained to derive information about critical conditions of the machine from the mined data of the sensors. Thereby desired quantities of a machine's operation that are crucial for efficiency or distinctive of failure can be derived, and an alert system may be realized. Whenever the machine operates inefficiently or failures are likely to occur, the alert system informs the operator, who then is able to initiate appropriate countermeasures. In data-driven CM, traditional ML algorithms, as well

as deep learning and classic neural networks, are applied [ZYW19]. Particularly supervised classification models are common for data-driven CM implementations. Therefore, data of different operation conditions need to be collected and annotated in advance in order to obtain a labeled data set. The data that is investigated in CM is naturally in chronological order. As a consequence, not only values of the recorded quantities contain information on a machine's condition, but also the chronological sequence of the measured values may contain information about a machine's condition. This circumstance leads to the problem of TSC, a prospering and demanded research field of the ML community.

3.3 Time Series Classification

The primary factor that sets TSC apart from other classification problems is the strict underlying order that the input data instances are subject to. The order typically stems from the temporal recording of the data at fixed intervals of time. Since class distinction may also lay in a sequence of quantities and not only in mere absolute values of quantities, TSC problems must be addressed differently compared to conventional classification problems. Given a set of n *time series*, $\boldsymbol{T} = \{T_1, T_2, ..., T_n\}$, where each time series has m ordered real-valued observations $T_i =< t_{i1}, t_{i2}, ..., t_{im} >$ and a class value c_n, the problem of TSC is to find a function that maps the ordered real-valued data instances, or predictors, to their respective class values [LDHB12]. Accordingly, a time series denotes an ordered sequence of data instances. Common fields where TSC problems typically occur are, among other things, genomic analysis, health informatics, finance, or human gait recognition [XPK10, TS12]. In general, every classification problem that includes a temporal component may be addressed as a TSC problem. According to [XPK10], TSC algorithms can be divided into three subcategories:

- *distance-based TSC*
- *model-based TSC*
- *feature-based TSC*

Distance-based approaches categorize data according to a distance measure. Time series which are globally similar to already classified time series are assigned to the category the similar and already classified time series belongs to. The *euclidean distance*

$$dist(s, s') = \sqrt{\sum_{i=1}^{m} (s[i] - s'[i])^2} \tag{3.1}$$

is a typical quantity to express global similarity between two time series s and s' of length m. A small distance indicates high similarity between two considered time series. Model-based approaches are based on generative models that model the joint probability distribution $P(X, Y)$ of time series X and a class Y. Consequently, new times series are classified to the category that yields the highest probability [XPK10]. Feature-based approaches classify a time series after its transformation into a feature vector. Here, every times series is characterized by a number of discrete variables which capture properties of the time series. These variables may be distinctive of *global properties* such as dominant frequencies, mean or standard deviation of the considered time series. Apart from that, also *local properties* such as meaningful patterns may be captured by appropriate features. Another benefit of the feature-based approach is its dimension-reducing nature as it transfers each time series into a smaller set of meaningful variables resulting in computationally less expensive procedures. The selected approach in this thesis is the feature-based TSC approach. In the next section, it is examined in more detail.

3.3.1 Feature-Based Time Series Classification

Considering a time series, two different domains occur, which are worthwhile to examine: The *time domain* and the *frequency domain*. The frequency-domain of a time series is obtained by a Fast Fourier Transformation (FFT) and may also contain meaningful information regarding a tackled TSC problem. Therefore, both domains should be considered within a TSC problem. In feature-based TSC, time series are first processed by a feature extracting process that aims to capture crucial properties of time- and frequency-domain. The key to a successful feature-based TSC is to find meaningful features capable of capturing crucial, class-distinctive properties of the time series. Besides mere similarity measures between time series as features, the statistical approach, which suggests deriving certain statistical quantities from a time series, is another widespread approach for feature-based TSC. Here, times series are distinguished by statistical characteristics in both time- and frequency-domain. Common statistical quantities utilized as time-domain features are, for instance, the mean μ or the standard deviation σ. In the frequency domain, quantities like the energy E of a FFT-transformed time series are popular [LHZ08]. Each time series here is considered as a discrete statistical distribution. The approach to utilize statistical characteristics is very common. Hence, it is used as a comparison measure for the method applied in this thesis. A detailed listing of statistical quantities used as features in this thesis is provided in section 4.4.1. However, since the statistical approach is rather suitable to capture global properties of a time series, class-discriminating information contained in local patterns of a time series may not be captured by this approach. One may, however, separate a time series into small subsequences manually and compute the mentioned statistical quantities in every manually created subsequence to derive locally distributed information of a time series. Nevertheless, one is initially blind regarding the position of meaningful patterns within a time

series and the length of these potential subsequences. To address these limitations, a new time series primitive, called *time series shapelets*, was introduced by [YK11]. Shapelets are additionally compatible with the frame of feature-based TSC since they can be used as features. The concrete meaning of shapelets and how they are used as features is clarified in the next section.

3.3.2 Time Series Shapelets

The idea of shapelets is to find subsequences in the available time series which are in some sense maximally representative of a class [YK11]. These subsequences are called shapelets and can be used as a discriminatory feature. A shapelet can be a subsequence of every possible length, up to the length of the original time series it stems from. Depending on a match or mismatch with a shapelet, a time series can be classified. Shapelets allow for TSC based on local, phase-independent similarity in shape [HLB+14]. That is, unlike the mentioned manual search for meaningful local regions in a time series, the method of shapelets is capable of performing a search for meaningful regions covering a whole time series, i.e. at every position. Additionally, shapelets depict a feature that captures only similarity in *shape*. Thus, potentially meaningful similarities in magnitude or position are *not* captured by shapelets. This is due to a prior normalization of subsequences used as shapelets and the phase-independent procedure of using shapelets as features. How shapelets are exactly utilized as features, how they are found, as well as problems and possibilities of this method, are outlined in the next part of this section.

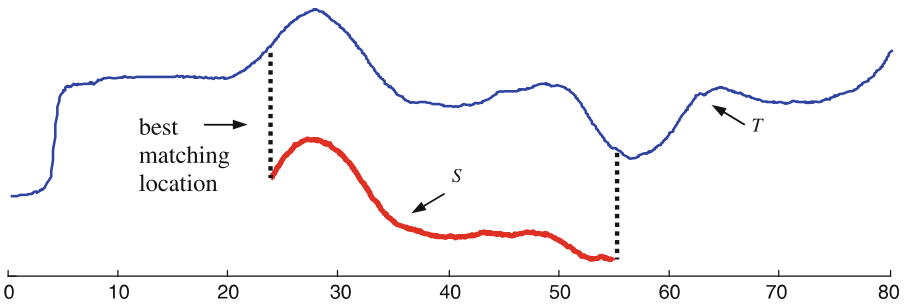


Figure 3.1: Process of sliding a shapelet along a time series to find the best matching position [YK11]

Considering a k-class problem of a set of n time series, $\boldsymbol{T} = \{T_1, T_2, ..., T_n\}$, all of length L and an already found set of d meaningful shapelets, $\boldsymbol{S} = \{S_1, S_2, ..., S_d\}$, then shapelets are utilized as features by computing the minimum distance of a shapelet to every time series in the data set (see Figure 3.1). The minimum distance allows to judge to what extent a shapelet is contained in a time series and thus to judge class membership. Since the length L of a time

series typically differs from the length l_s of a shapelet, the minimum distance of shapelet and time series is found by sliding the shapelet across the whole time series and measuring the distance between both for every position, i.g. from start to end of the time series. Thereby, a set of distances between time series and shapelet is obtained. The minimum distance of this distance set is then used as the respective feature value of a shapelet for a considered time series. In the literature, several approaches are introduced to find meaningful shapelets in a set of time series. Yet, there are general steps within the shapelet search that can be found in many contributions:

1. *Shapelets are extracted by scanning the available time series data set.*
   However, different approaches exist in how to find shapelets in the data set effectively. [YK11] and [HLB$^+$14] perform an exhaustive search to find shapelets. That is, for every time series in the set, every possibly extractable subsequence of every position and every possible length is considered as a possible *shapelet candidate*. Since this is very time-consuming and computationally expensive, [MKY11] and [RK13] present each a search method that effectively prunes the search space within the time series while maintaining a comparable quality of the found shapelets. In fact, a major part of conducted research in the field of shapelets attempts to reduce the complexity of algorithms for shapelet discovery [GWST16, MKY11, RK13, Lu 16].

2. *Shapelets are chosen regarding their discriminatory power.*
   The majority of the proposed methods of finding shapelets first extract potential candidates and evaluate them regarding their discriminatory power afterward. That is, the extent to which a candidate is characteristic for class membership is measured. [YK11] apply *information gain* to measure the discriminatory power of shapelets. [HLB$^+$14] instead study different statistical quantities to measure discriminatory power and conclude that *F-statistic* delivers the best shapelets for multi-class problems. After extracting and evaluating a specific number of shapelet candidates, the algorithm returns a list containing the best class-distinctive shapelets.

3. *Shapelets are inspected regarding self-similarity.*
   Time series data sets contain naturally similar time series since they are usually samples of a certain joint event. Moreover, similarity among time series of the same class increases, intuitively, as well. Under this assumption, well class-distinguishing shapelets are likely to be similar to each other when being characteristic of a class [GWST16]. Since shapelets are normalized subsequences of the original time series, the probability that these patterns occur several times in the data is very high. This is due to the manner in how discriminatory power of shapelets is considered. A high-quality shapelet matches time series of a certain class and does not fit in time series from other classes. That is,

a high-quality shapelet is also contained in the majority of the time series, which stem from the same class the shapelet is characteristic of. Consequently, this shapelet is found frequently during shapelet discovery. One would, however, prefer independent shapelets as features because similar shapelets contain similar information and thus are inefficient. Similar shapelets may even diminish accuracy in case other well-discriminating shapelets are neglected given a fixed number of shapelets as features for a classifier [HLB+14]. Therefore, several papers in the literature propose methods to measure similarity between already found shapelets and accept or reject a shapelet after a quality comparison with other similar shapelets. E.g., [HLB+14] suggest a clustering-shapelets-algorithm after shapelet discovery that aims to reveal independent shapelets. [GWST16] embed a similarity filter in the shapelet discovery and prune low-quality candidates that are similar to already considered candidates.

Furthermore, when applying the shapelets method as features for classification, the following assumptions, problems, and opportunities are incorporated:

- Using shapelets as features automatically raises the assumption, the tackled TSC problem is solvable by similarity in shape. In other words, the class membership of time series is defined by the shape of local patterns. Other potentially meaningful features like mere magnitude as well as the position of meaningful patterns are not captured by shapelets. Recalling that shapelets are *normalized* subsequences of a time series, mere presence or absence of *shapes* within a time series define class membership according to the shapelet method. This might be problematic since rules that define class membership are initially unknown when tackling a TSC problem.

- Shapelets are a promising possibility to improve the interpretability of a classifier and gain further insights into the data. Since shapelets are extracted by available time series of the problem, a review on the best discriminating shapelets after classification enables a reasonably simple interpretation of the result [YK11]. Shapelets reveal parts of a time series, that define its class membership and therefore promote interpretability.

4 Condition Monitoring of Toothed Belt Drives

In this chapter, a CM for toothed belt drives is designed. Therefore, a classifier is trained to detect the tension of the considered toothed belt. For training, each time series is transformed into the *shapelet space* after an extraction and selection of meaningful shapelets. A suitable algorithm that performs shapelet discovery and extraction is to be selected previously. Furthermore, to enable a robust evaluation of the shapelet method, an additional classifier is trained with *handcrafted, statistical* features. The different steps of selecting and extracting features as well as transforming every time series into the feature space are outlined in more detail. All applied algorithms are presented and explained. Prior to the design of a CM system, some general information about toothed belt drives as well as a description of the required data acquisition and the related test facility are provided.

4.1 Toothed Belt Drives

Toothed belt drives are drives that transmit power via a rubber belt that is equipped with teeth on its active side, i.g. the side that interacts with the pulleys. For the sake of uniform terminology with the literature, toothed belt drives are referred to as *timing belt drives* in the following. In general, timing belt drives are drive systems that comprise a timing belt and two or more timing pulleys (see Figure 4.1) [PO12]. During operation, the timing belt transmits power from a driving pulley to a driven pulley by meshing into the toothed pulleys. A favorable feature of timing belts is their ability to transmit motion synchronously [KBG+16]. Their motion transmissions always have a uniform ratio due to the teeth meshing into the toothed pulleys. Hence, no slip is involved. Therefore, timing belts are popular drives applied in modern stepper motors and servo technology since they enable precise positioning [PO12]. Additionally, they combine the advantages of high permissible speeds with low-noise operation as stated by [PO12]. Another benefit of timing belts is their potentially high load capacity due to the belt's large number of simultaneously meshing teeth [KM88]. During operation, one side of the belt, the loaded side, is subject to higher tension compared to the idle side. However, reliable power transfer is only possible as long as residual tension remains in the idle side of the belt. Hence, the timing belt should be pre-tensioned [PO12, Feh19]. Though, too high tension results in an unnecessary preload of involved shafts and bearings, whereas too low tension causes unwanted jumping of the timing belt [KM88]. The tension is set by applying a specific pre-tension force

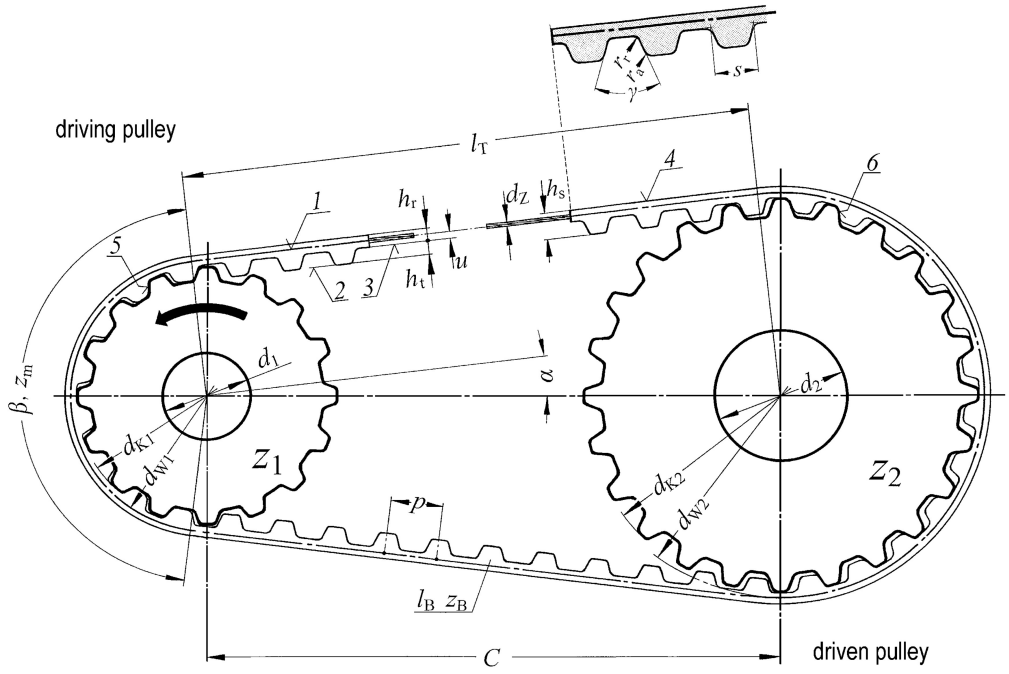


Figure 4.1: Composition of a timing belt drive with driving pulley and driven pulley [PO12]

F_V to the belt. Therefore, an optimal pre-tension force applied to the timing belt is to be set and maintained in order to guarantee efficient operation.

4.2 Test Facility and Data Acquisition

To enable training a classifier, first appropriate data need to be acquired. Previously to this work, other investigations regarding a robust data-driven estimation of the tension of toothed belt drives have already been conducted. [Feh19] has designed a CM for toothed belt drives by extracting meaningful statistical features of time series that stem from recordings of attached internal sensors. Features were extracted from both time and frequency-domain. Subsequently different classification models were trained and evaluated. Therefore, this work can be seen as an addition to [Feh19]. Accordingly, appropriate data sets of the considered toothed belt drive's operation have already been prepared and are provided for the investigations in this thesis. A detailed description of the data acquisition process and the required annotation of data is therefore not included in this thesis but can be obtained by inspecting [Feh19]. Merely a general presentation of the used data sets and their labels, as well as a brief description of the related test facility, are provided.

To investigate data-driven approaches of estimating the tension of operating timing belt drives,

a testing facility was assembled at the *Institute of Mechatronic Systems* (IMES), Hannover (see Figure 4.2). The assembly comprises a drive pulley, a load pulley, a tensioner pulley, a linear potentiometer, the timing belt itself, and a controller. Drive and load pulley are each driven by a servo motor. The two motors can be excited in opposite directions so that a load to the drive is simulated. The pre-tension force applied to the timing belt can be adjusted by moving the tensioner pulley towards or off the timing belt. By this, different tension configurations are created. The pre-tension force F_{V} is indirectly obtained by the measured voltage across the linear potentiometer whose electrical resistance is proportional to the position of the tensioner pulley and thus to the pre-tension force. The connected controller allows to specify requested trajectories for the test facility. Meaningful data is acquired by taking measurements of test

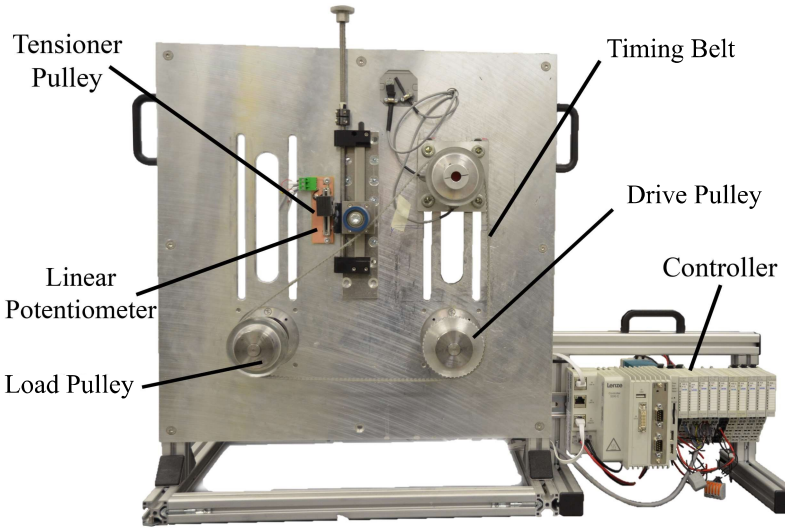


Figure 4.2: Test facility at IMES, Hannover [Feh19]

runs of the test facility and successively changing the applied pre-tension force to the timing belt. The drive motor is excited by the controller so that it executes the requested trajectory. To create comparable time series for a data set, a fixed trajectory is adjusted at the controller so that changes in the time series are likely to be meaningful for the tension. The conducted test runs persist only for a short time, namely a few tenths of a second. All time series are measured equally from recordings of the drive motor. In this thesis, two different data sets are investigated that differ in the specified trajectory. The first data set contains measurements of test runs subject to a *jerk-limited trajectory* (JLT). The second one comprises time series from test runs subject to a *multi frequency excitation* (MFE). All time series within a data set stem from test runs which are all equally excited. To each time series of a data set, the respective measured pre-tension force is also attached so that a data set contains both time series and the pre-tension force as target variable. For both data sets, two subsets were independently created.

One larger subset is intended for model training and subsequent hyper-parameter optimization. The smaller subset is intended to obtain a final evaluation of the created classifier with its selected hyper-parameters.

4.2.1 Data Description

For the JLT data set (see Table 4.1), acceleration, position error, temperature, torque, set-velocity, and the velocity itself are recorded from the drive motor. The position error is obtained by comparing the target trajectory specified by the controller with the actual measured trajectory of the drive motor. For all test runs, an approximately equal temperature is maintained to exclude temperature influences on the acquired data. The recording of the data is carried out with a sampling rate of 1 kHz resulting in a total of 483 temporally equidistant data samples per time series. All signals are subsequently transformed into the frequency domain via a FFT so that each measurement consists of 12 time series in total. In summary the JLT data set consists of $6 \times 1099 \times 483$ data samples in time-domain and $6 \times 1099 \times 242$ data samples in frequency-domain.

Table 4.1: JLT Data set

Variable	Time	Frequency	Total Size	Train/Test	Sampling Rate
Acceleration	✓	✓	1099	899/200	1 kHz
Position Error	✓	✓	1099	899/200	1 kHz
Temperature	✓	✓	1099	899/200	1 kHz
Torque	✓	✓	1099	899/200	1 kHz
Set-Velocity	✓	✓	1099	899/200	1 kHz
Velocity	✓	✓	1099	899/200	1 kHz

The MFE data set (see Table 4.3) consists of time series that stem from a MFE-run. Here, a specified multi-frequent torque is inserted to the motor. Like the JLT data set, it is also recorded with a sampling rate of 1 kHz, but instead it is composed of 512 temporally equidistant data samples for each time series. This data set consists only of time series that display the predefined torque and the resulting velocity of the drive motor. Since the torque is predefined, it is equal for every test run so that only measurements of the velocity contain meaningful information. Here as well, the signals are transformed into the frequency domain via a FFT. In total, the MFE data set consists of $2 \times 1099 \times 512$ data samples in time-domain and $2 \times 1099 \times 256$ data samples in frequency-domain.

Table 4.3: MFT Data set

Variable	Time	Frequency	Total Size	Train/Test	Sampling Rate
Torque	✓	✓	1099	899/200	1 kHz
Velocity	✓	✓	1099	899/200	1 kHz

4.2.2 Data Labeling

Each data set contains time series from test runs subject to different pre-tension forces. The scope of considered pre-tension forces ranges for both data sets approximately the same. For the JLT data set, the pre-tension force ranges from $F_{\mathrm{V_{JLT}}} = 0\,\mathrm{N}$ up to $F_{\mathrm{V_{JLT}}} = 201{,}87\,\mathrm{N}$. For the MFE data set, the minimum value is $F_{\mathrm{V_{MFE}}} = 0\,\mathrm{N}$, whereas the maximum is observed at $F_{\mathrm{V_{MFE}}} = 195{,}76\,\mathrm{N}$. According to calculations from [Feh19], the nominal pre-tension force for the test facility's timing belt is

$$F_{\mathrm{V,nominal}} = 83{,}9\,\mathrm{N}. \tag{4.1}$$

In order to design a classifier, a discrete target variable must be created from the measured pre-tension force. Therefore, target values are assigned into evenly distributed intervals around the nominal pre-tension value. After normalizing each target value with respect to the nominal value, so that $F_{\mathrm{V,nominal,norm}} = 1$, the following categories

$$y_{\mathrm{categorical}} = \begin{cases} 1\text{: } \textit{"Critically Loose"}, & \text{if } F_{\mathrm{V,norm}} = 0 \\ 2\text{: } \textit{"Loose"}, & \text{if } 0 < F_{\mathrm{V,norm}} \leq 0.5 \\ 3\text{: } \textit{"Normal"}, & \text{if } 0.5 < F_{\mathrm{V,norm}} \leq 1.5 \\ 4\text{: } \textit{"Tight"}, & \text{if } 1.5 < F_{\mathrm{V,norm}} \leq 2 \\ 5\text{: } \textit{"Critically Tight"}, & \text{if } F_{\mathrm{V,norm}} > 2 \end{cases} \tag{4.2}$$

are defined for a 5-class problem. Discretizing the pre-tension force into five categories is a good compromise between obtaining enough detailed information on the pre-tension force and still preserve enough data samples of each category for training. After a successful discretization of the target variable, the different classes in the JLT dataset are distributed as displayed in Figure 4.3.

4.2.3 Data Exploration

To provide a first understanding of the data, some insights into the intra-class variance of the different time series contained in each data set are presented. This section provides first a

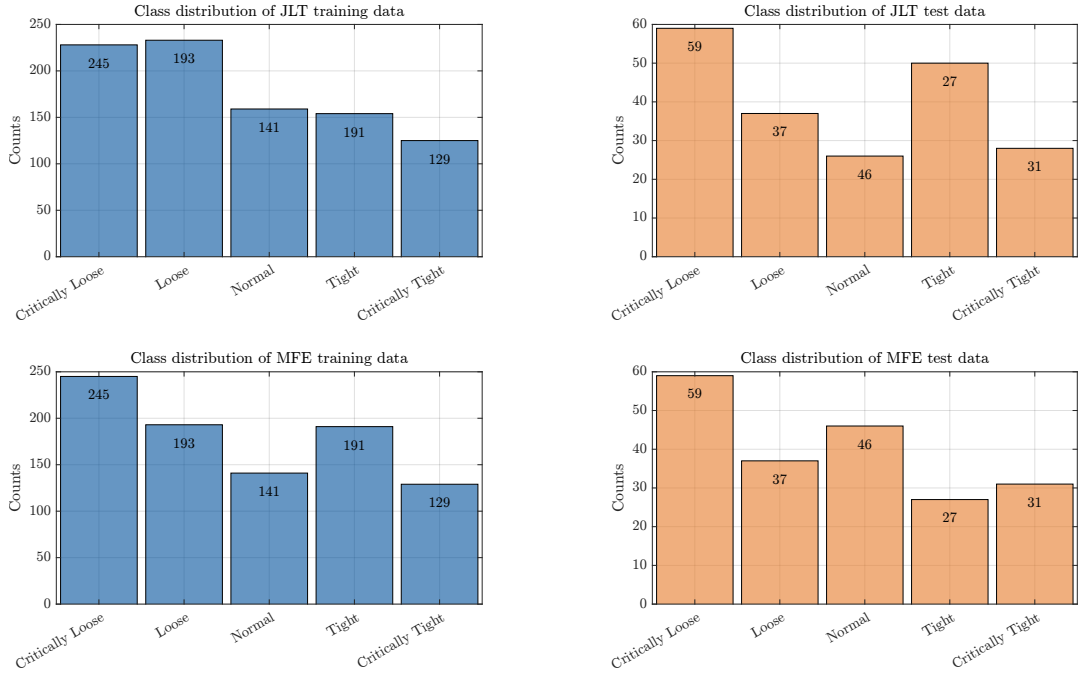


Figure 4.3: Class distribution of the JLT data set and the MFE data set

discussion on the JLT data set and concludes with a data exploration of the MFE data set.

To depict intra-class variability, initially, meaningful time series of the JLT data set are sorted by class, after which the mean of each group is plotted. Figure 4.4 displays the mean of each class group in time and frequency-domain. Here, promising variables for class distinction by shape in time-domain seem to be PositionError and Torque. For the plots of Velocity and Acceleration, only slight differences between classes are visible. Since Temperature and SetVelocity show no intra-class variability at all, they are worthless for TSC and thus are not displayed. Considering the plots in frequency-domain, Acceleration, Torque, and Velocity show intra-class variability and should be investigated.
For the MFE data set, only measurements of Torque and Velocity are available. Since the drive motor here is excited with the same torque curve for every test run, no intra-class variability of torque time series exisits. Plotting the class mean of the measured Velocity in time-domain (see Figure 4.5), however, indicates meaningful intra-class variability. Therefore, Velocity seems promising for class distinction by similarity in shape. In Figure 4.5 the mean of each class group in frequency-domain is also plotted. Again, promising intra-class variability is visible and the data seem to be promising.

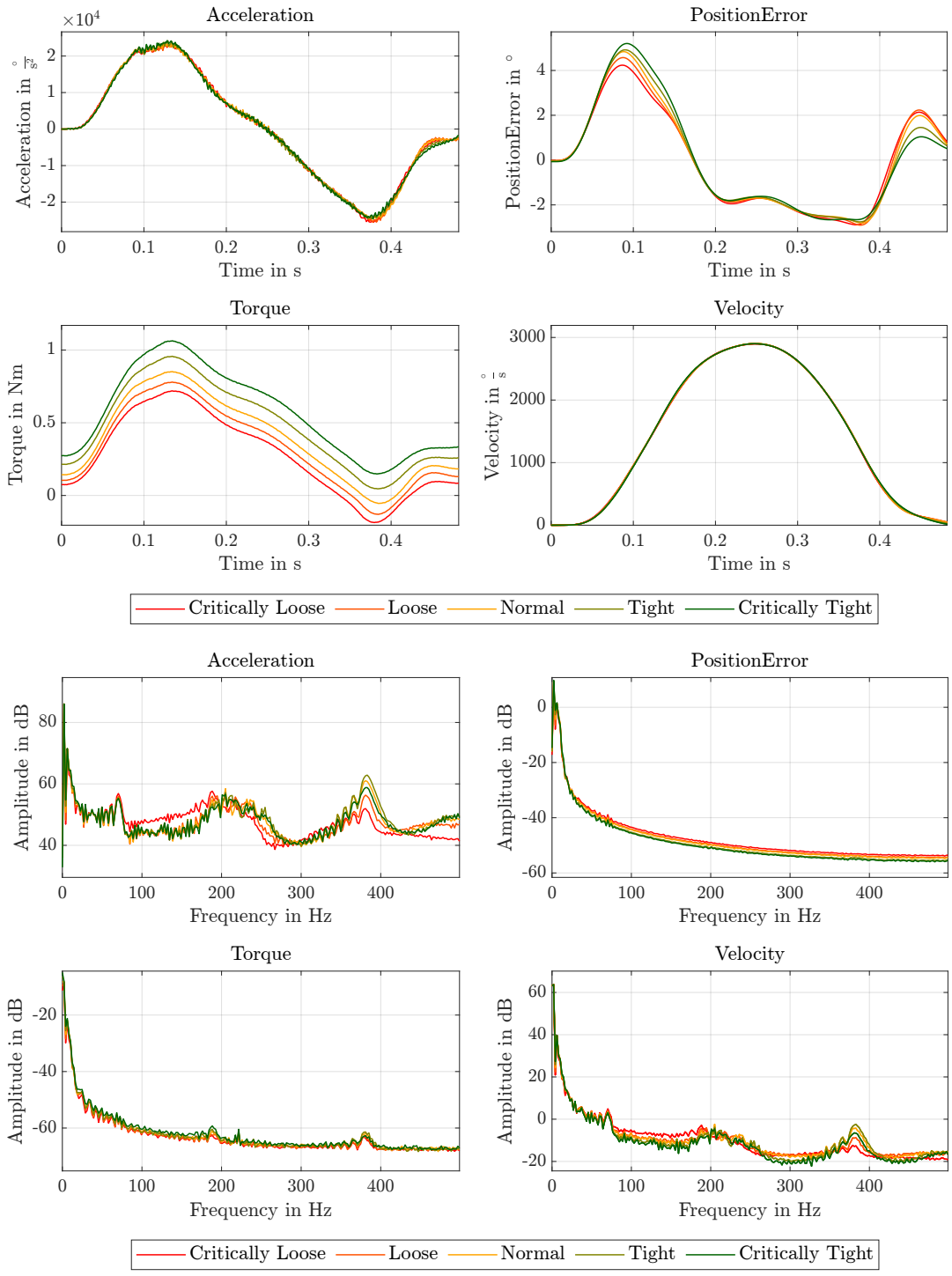


Figure 4.4: Class-average time- and frequency-domain plots of JLT data

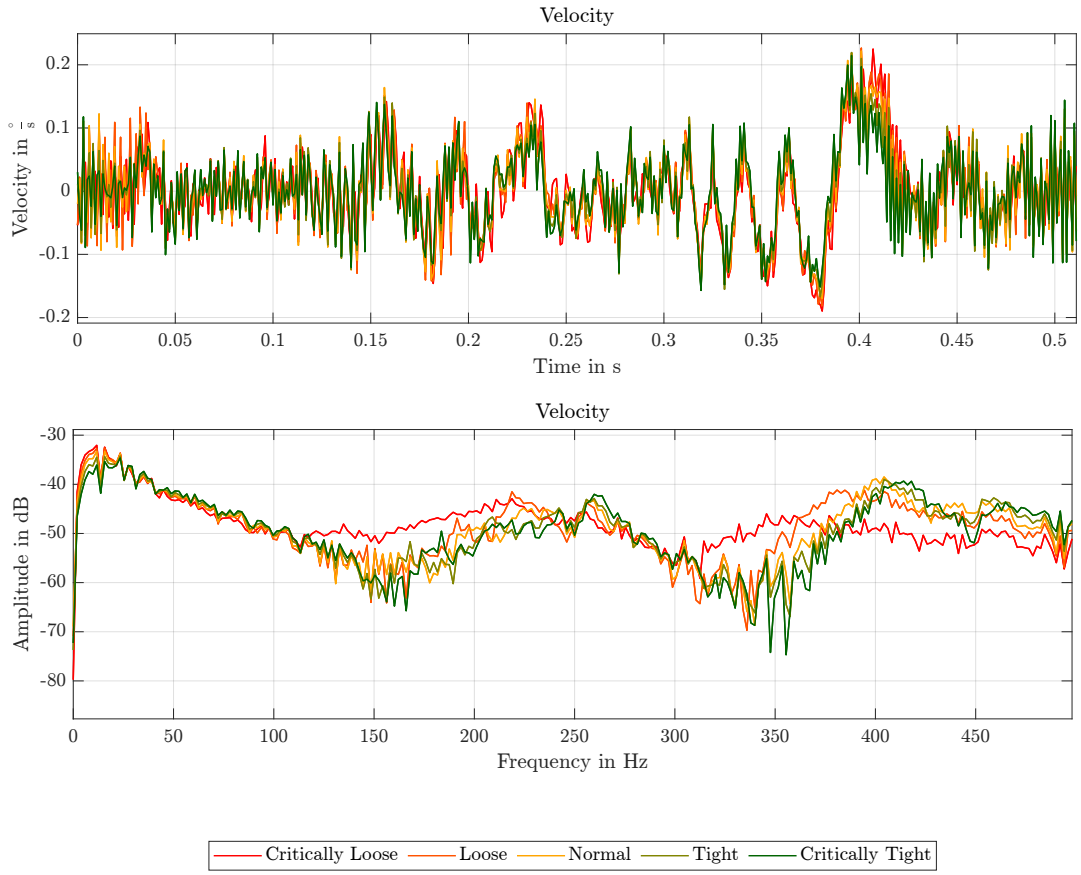


Figure 4.5: Class-average time- and frequency-domain plots of MFE data

4.3 Feature Extraction based on Shapelets

In order to create a classifier based on shapelets, first an appropriate algorithm for shapelet discovery must be selected. In order to facilitate the decision of which algorithm will be selected, some objectives for the shapelets search are defined.

4.3.1 Objectives

1. *Extracted shapelets should be as meaningful as possible to enable a competitive comparison to the statistical approach*
   Consequently, the selected algorithm should conduct a comprehensive search that aims to detect the best shapelets in a data set.

2. *Extracted shapelets should be compatible to the frame of feature-based TSC*
   That is, the selected algorithm should not embed the shapelet search in an already previously picked classifier so that it evaluates shapelets only based on their performance on a

predefined classifier. Rather, shapelets search should be independent of the subsequent construction of a classifier so that shapelet assessment is generally valid and not only for a given classifier. This objective results from the task of comparing the shapelets approach with a TSC based on statistical features. A meaningful comparison is only possible when both approaches use the same classifier. Therefore, shapelets should be obtained in such a way that they are acceptable for many classification models.

3. *Shapelets should be stored for the subsequent evaluation regarding interpretability*
   During shapelet search, the possibility to store the raw shapelets and not only their distances to other time series should be given.

4.3.2 Selection of a Shapelet Discovery Algorithm

In the literature, the first published papers in the field of time series shapelets all propose a discovery that is embedded in a simultaneous construction of a decision tree [YK11, MKY11, RK13]. Apart from that, the majority of all proposed methods aim to reduce the time complexity of the shapelets search since it appears to be the main weakness of the first publications in this regard. By doing this, several rather complex procedures were suggested to prune the shapelet search significantly [GWST16, Lu 16]. However, minimum time complexity is not a prime objective in this thesis which is why it is neglected in the frame of algorithm selection. [HLB$^+$14] propose the first shapelet discovery algorithm that does not embed the shapelet search in constructing a preselected classifier. In fact, they aim to establish shapelets as a general time series primitive and suggest to transform time series data into a *shapelet-space*. That is, after a successful discovery of meaningful shapelets, every times series is transformed into its respective distances to the extracted shapelets. [HLB$^+$14] call this procedure *shapelet transformation* of the time series data. Additionally, the general objective of this algorithm is to find the k-best shapelets in a time series data set. In this regard, it matches the first previously defined objective of algorithm selection. Furthermore, this procedure is well compatible to the frame of feature-based TSC, since the distance sets are acceptable to a wide collection of classifiers and can be considered as features. Thus, with this procedure, a fair comparison to the statistical feature-based approach from [Feh19] is possible, and the second objective is met as well. In addition, this approach is, compared to the mentioned algorithms, rather simple and enables a simple storing of extracted shapelets. The great time complexity of this algorithm is acceptable considering its rather simple implementation and the resulting flexibility of being able to select any desired classifier afterward. Therefore, the algorithm of [HLB$^+$14] is applied in this thesis to extract shapelets and make use of them as features for training a classifier.

In general, the method of [HLB$^+$14] can be divided into three successive steps:

1. *Shapelet discovery and extraction*
2. *Filter shapelets*
3. *Shapelet Transformation*

The following sections present each step in detail and explain the related algorithms. All algorithms were implemented in MATLAB as well as all experiments were conducted in MATLAB.

4.3.3 Shapelet Extraction

As mentioned, the method of [HLB+14] aims to generate a list of k-best shapelets sorted with respect to their discriminatory power. Therefore [HLB+14] propose an exhaustive search to detect the best shapelets. To provide a comprehensive insight into the algorithm, first, some terms and definitions related to the shapelet discovery, that also occur in [HLB+14], are presented:

1. A data set $\mathbf{T} \in \mathbb{R}^{n \times m}$ with n time series of length m is called *time series data set*.
2. A time series of length m consists of m data samples:

$$T_{i,l} =< t_{i,1}, t_{i,2}, ..., t_{i,m} > \tag{4.3}$$

3. A *shapelet* is a normalized subsequence of one time series in a data set $\mathbf{T}$.
4. A set of all normalized subsequences of length l for time series T_i is denoted as $W_{i,l}$. This set contains $(m - l) + 1$ different candidate shapelets of legth l. The set of all subsequences of length l for a time series data set $\mathbf{T}$ comprises:

$$W_l = \{W_{1,l} \cup W_{2,l} \cup ... \cup W_{n,l}\}. \tag{4.4}$$

5. Let l range between min and max, so that $min \leq l \leq max$ is valid. Accordingly, the set of all candidate shapelets for the data set $\mathbf{T}$ contains:

$$\mathbf{W} = \{W_{min} \cup W_{min+1} \cup ... \cup W_{max}\}. \tag{4.5}$$

6. The set $\mathbf{W}$ is composed of

$$|\mathbf{W}| = \sum_{l=min}^{max} n(m - l + 1) \tag{4.6}$$

candidate shapelets.

7. The squared euclidean distance between two subsequences S and R, both of length l, is obtained by

$$dist(S, R) = \sum_{i=1}^{l} (s_i - r_i)^2. \tag{4.7}$$

8. The distance between a subsequence S of length l and a time series T_i is the minimum distance between S and all normalized length l subsequences of T_i and is calculated by

$$d_{S,i} = \min_{R \in W_{i,l}} dist(S, R). \tag{4.8}$$

Obviously, when calculating the distance between two subsequences, both need to be in the same dimension. That is why all successively compared subsequences of a time series with a shapelet are previously normalized.

9. Computing the distances of a shapelet candidate S to every time series T_i of a data set $\mathbf{T}$ results in a set of n distances:

$$D_S =< d_{S,1}, d_{S,2}, ..., d_{S,n} > . \tag{4.9}$$

Algorithm Procedure

The procedure of shapelet extraction encompasses three general steps: First, shapelet candidates $\boldsymbol{W}$ are generated, then for each candidate, a distance set D_S is calculated. Finally, an assessment of each shapelet candidate based on the distance set D_S is conducted, and shapelet candidates are sorted regarding discriminatory power in descending order. The detailed procedure of the extraction algorithm (see Algorithm 1) from [HLB$^+$14] is as follows:

At the start, a list that will contain the k-best shapelets and their respective assessment values is initialized. Thereon, an iteration through every time series in the data set is carried out. For each time series, shapelet candidates of every possible length between min and max of every possible position within the time series are generated. Considering a time series that is to be searched for shapelets of length l, candidates are generated by sliding a window of length l across the time series. For every step the window takes, a shapelet candidate is extracted defined by the position and length of the window. Thus, $m - l + 1$ shapelet candidates from time series of length m are generated and stored in a set $W_{i,l}$. Then, for each shapelet candidate of that set, a distance set D_S containing distances to every time series of the data set is generated by $findDistances(S, \mathbf{T})$. Subsequently, every shapelet candidate

is assessed regarding discriminatory power by a specified function $assessCandidate$. The function quantifies to what extent the resulting distribution of a shapelet candidate's distances to all time series is class-distinctive. After all generated shapelet candidates of a considered time series are assessed, they are sorted by quality in descending order. Thereafter, the list is filtered regarding self-similarity. The filtering regarding self-similarity is due to the comprehensive manner candidates are extracted from a time series. Since shapelet candidates are extracted from every possible position within a time series, covering every possible length of that position, a lot of similar candidates occur. In fact, candidates that are, apart from one data sample, exactly the same, are very frequent. Hence, shapelet groups with mutual indices are filtered so that only the best-discriminating shapelet candidate of that group is preserved. The rest is discarded. Eventually, the list of k-best shapelets is updated after a competitive comparison between the best-extracted shapelet candidates of the considered time series and the already existing k-best shapelets. This procedure is repeated until all time series have been examined.

Algorithm 1 ShapeletCachedSelection($\mathbf{T}, min, max, k$) according to [HLB+14]

```
1:  kShapelets ← ∅
2:  for all T_i in T do
3:      shapelets ← ∅
4:      for l ← min to max do
5:          W_{i,l} ← generateCandidates(T_i, l)
6:          for all subsequences S in W_{i,l} do
7:              D_S ← findDistances(S, T)
8:              quality ← assessCandidate(S, D_S)
9:              shapelets.add(S, quality)
10:         end for
11:     end for
12:     sortByQuality(shapelets)
13:     removeSelfSimilar(shapelts)
14:     kShapelets ← merge(k, kShapelets, shapelets)
15: end for
16: return kShapelets
```

Evaluation Metric of Shapelets

The paper of [HLB+14] conducts, besides the presentation of their shapelet discovery algorithm, a study on different assessment metrics for shapelet quality. They conclude that *F-statistic* appears to be the best evaluation metric when tackling multi-class problems. Therefore, F-statistic is applied in this thesis to evaluate a shapelet candidate's discriminatory power. It is involved in the $assessCandiate$-function of Algorithm 1. The F-statistic metric quantifies

how well shapelets are class-distinctive by putting the intra-class variance of the resulting distance distribution in relation to the inter-class variance of the distance distribution. That is, the obtained distances of a shapelet are first sorted by class membership so that for each class, a group of distances is obtained. After that, the variance within each group of distances, the inter-class variance, is computed. Additionally, the intra-class variance is obtained by first averaging the distances of each group and second measuring the variance of the obtained means. F-statistic is defined as follows:

$$F = \frac{\sum\limits_{i} \left(\bar{D}_i - \bar{D}\right)^2 / (C-1)}{\sum\limits_{i=1}^{C} \sum\limits_{d_j \in D_i} \left(d_j - \bar{D}_i\right)^2 / (n-C)}. \tag{4.10}$$

$\bar{D}_i$ is the mean of the distance set belonging to class i, whereas $\bar{D}$ is the overall mean of the whole distance set. C is the number of classes and n the number of time series in the data set.

Algorithm Modifications and Parameter Adjustment

Since the involved data sets of this work are very large, some modifications to Algorithm 1 are made to preserve a feasible run time. The JLT training data set consists of $6 \times 899 \times 483$ data samples in time-domain and $2 \times 899 \times 242$ in frequency-domain. In this thesis, shapelet discovery is only applied to the training data set. Within the study of [HLB$^+$14], the largest data set that is exposed to shapelet discovery according to Algorithm 1 comprises $n = 400$ time series of length $m = 250$. [HLB$^+$14] observe that shapelet discovery for this data set lasts 37.4 hours. Furthermore, it is determined that the time complexity of Algorithm 1 is $O(n^2 m^4)$ [HLB$^+$14]. Predicting the run time of shapelet discovery for one variable of the JLT data set results accordingly in a duration of 2822.5 hours, which is clearly infeasible. Therefore some parameters of Algorithm 1 are modified as follows:

1. For every time series, the set of shapelet lengths to be considered is constrained. Since the most promising lengths for shapelets are initially unknown, a comprehensive search that comprises both very small lengths as wells as lengths covering almost the whole time series is favorable. Therefore, in this thesis, shapelet lengths are successively picked from an equidistantly discretized interval with an increment of 10. For a time series of length $m = 483$, the interval that contains all shapelet lengths to be considered is defined as follows:

   $$I_l = [10, 20, 30, ..., 400]. \tag{4.11}$$

   Additionally, to evaluate if local patterns are superior to global distances as features for the given TSC problem, eventually, the whole times series, i.g. shapelets of length

$l = 489$, are considered and assessed.

2. To decrease the time complexity further, another modification would be to constrain the number of positions within each time series from which candidates are extracted as possible shapelets. In this thesis, not from every possible position within a time series candidates are extracted, as proposed by [HLB+14]. Rather, positions are defined in a way that no overlapping candidates, given a specific candidate's length, are extracted (see Figure 4.6). By that, the whole time series is still searched for shapelets, but the number of total shapelet candidates per time series is reduced.

3. The final modification aims to speed up distance calculations of $findDistances(S, \mathbf{T})$. Therefore, a proposal of [MKY11] is applied to compute distances more efficiently. To determine the distance between a candidate S of length l and a time series T_i, the euclidean distance between every subsequence of T_i of length l and the candidate S is computed, and the minimum is inserted into D_S. During shapelet discovery, a lot of similar candidates are inspected and assessed. Therefore, a lot of redundant calculations, especially during distance computations, are performed. To mitigate this, [MKY11] propose to store certain statistical quantities between two time series that help to reduce the number of required computations to find distances between candidate S and time series T_i. Based on these previously computed statistics, a different distance metric is applied by [MKY11]. They find distances between a candidate S and a subsequence X of T_i based on the covariance between both. The distance is obtained by

$$dist(S, X) = \sqrt{2(1 - C(S, X))} \tag{4.12}$$

with the covariance

$$C(S, X) = \frac{\sum SX - l\mu_S\mu_X}{l\sigma_S\sigma_X}. \tag{4.13}$$

$\sum SX$ is the sum of products between both subsequences, l the length, and μ and σ the mean and deviation of both subsequences. The quantities that are stored before distance calculations help to find $\sum SX$, μ_S, μ_X, σ_S, and σ_X in constant time. For more details of this approach, see [MKY11].

After a full run of Algorithm 1 a sorted list with k-best shapelets and the related F-statistic values is obtained. During that procedure, shapelet candidates extracted from the same time series are filtered regarding self-similarity by $removeSelfSimilar(shapelts)$. However, self-similarity between shapelets of different time series is not inspected. [HLB+14] state that the k-best shapelets obtained from Algorithm 1 are still subject to high self-similarity since the best discriminating shapelets are contained in many time series. Therefore, a filtering process of

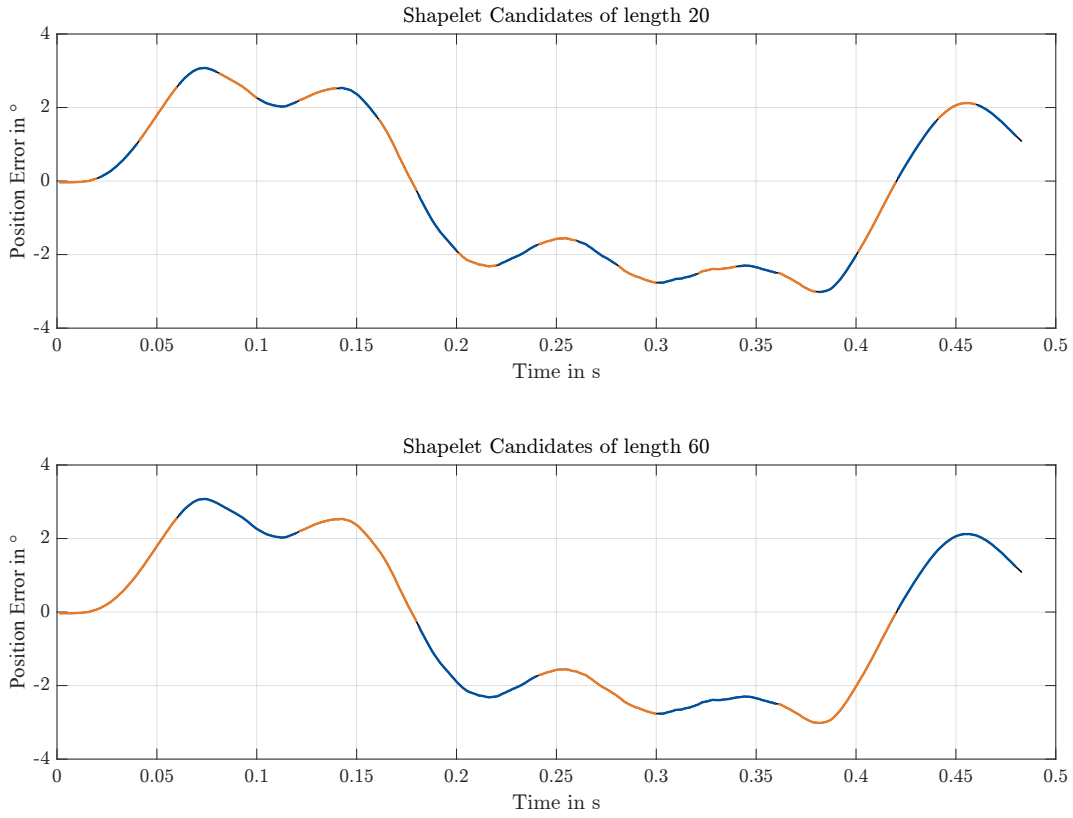


Figure 4.6: Exemplary illustration of shapelet discovery that avoids overlapping indices

obtained shapelets is carried out to reveal independent shapelets and thus enrich the information density of obtained features.

4.3.4 Filter Shapelets

To measure self-similarity of shapelets stemming from different time series, again, a distance metric is applied. Top-k shapelets are filtered by computing a distance matrix $\mathbf{M}$ that contains the distances of each shapelet to every other shapelet. Thereon, the pair that is most similar to one another is compared regarding shapelet quality. Of this pair, the shapelet that is of lower quality is subsequently discarded. Consequently, the set of k-best shapelets shrinks after a comparison. This process of comparing is repeated until a predefined size n_{filter} of the shapelet set is reached, and the algorithm stops. Eventually, a set of independent shapelets is revealed. Initially, it is unknown how many meaningful independent shapelets exist for a given TSC problem. Thus, it is favorable to try out different numbers n_{filter} to obtain the most efficient configuration of a feature set that is built up of shapelets.

4.3.5 Shapelets as Features

Finally, to make obtained shapelets accessible to classification algorithms, distances between a shapelet and every time series of a data set are calculated (see Algorithm 2). [HLB$^+$14] call this *shapelet transformation* since the time series data set is transformed into a set of distances to shapelets. Accordingly, every time series is described by a set of distances to meaningful shapelets and can be classified based on these values. Distance values are a meaningful measure to state whether a shapelet is contained in a time series or not. After the data transformation, standard procedures of supervised learning and conventional classification algorithms can be applied to the data.

Algorithm 2 ShapeletTransform(Shapelets S, Dataset $\mathbf{T}$) according to [HLB$^+$14]

```
T' ← ∅
for all T_i in T do
    for all shapelets s_j ∈ S do
        dist ← findDistance(s_j, T_i)
        T_{i,j} = dist
    end for
    T' = T' ∪ T_i
end for
return T'
```

4.3.6 Shapelet Subset Selection

In this thesis, Algorithm 1 is carried out with a value of $k = 200$. [HLB$^+$14] use $k = \frac{m}{2}$, where m is the length of time series from a data set. $k = 200$ is not exactly half the length m of the considered time series in this thesis, but it is near that value. Furthermore, the number k does not necessarily determine the performance of shapelet discovery, but it should be chosen high enough. After a full run of shapelet discovery, for every variable in time and frequency-domain, a list containing the 200-best shapelets is generated. Recalling that the JLT data set consists of 4 promising variables, 1600 shapelets would be obtained by a full run. In order to obtain a more efficient feature set and to concentrate independent shapelets, all lists are filtered individually by proceeding as described in section 4.3.4. This procedure is also important to facilitate an interpretation of the obtained shapelets afterward since fewer shapelets are easier to interpret. The filtering procedure of section 4.3.4 is subject to a hyperparameter that determines to how many shapelets the set should be reduced. To evaluate how many independent shapelets of each variable exist, the filter process is carried out several times with a successive increment of the number n_{filter} that determines to how many shapelets the list is filtered. With each filtered list of shapelets, a random forest is trained and evaluated several times using 10-fold

cross-validation. Since random forests are rather unaffected by bad features or redundant information, it is assumed that the accuracy increases with an increasing number of shapelets to a certain point. After that point, saturation should be reached. The idea is to determine the number of independent shapelets of each variable by determining the saturation point, of which a number of $n_{\text{independet}}$ shapelets per each variable can be derived. The range of n_{filter} for all experiments in this regard is

$$5 \leq n_{\text{filter}} \leq 200. \tag{4.14}$$

For each filter process the number n_{filter} is increased by 5 so that the set $\mathbf{N}$ of trials is

$$\mathbf{N} = \{5, 10, 15, ..., 195, 200\}. \tag{4.15}$$

The result of this experiment is displayed in Figure 4.7. The number of shapelets is plotted against the normalized obtained accuracy. For most variables, even a small number of shapelets already yield good accuracies compared to the maximum accuracy. Since mostly no significant increases in accuracy are visible, the number n_{filter} in this thesis is set at the number that first exceeds or equals a relative accuracy of $ACC_{\text{rel}} \geq 0.9$. Accordingly, the shapelet-feature subset as displayed in Table 4.5 is used in this thesis to generate benchmark classifiers for all further experiments.

Table 4.5: Shapelet subset selection of JLT and MFE data

JLT			MFE		
Variable	Domain	Shapelets	Variable	Domain	Shapelets
Acceleration	Time	40	Acceleration	Time	×
Position Error	Time	10	Position Error	Time	×
Torque	Time	10	Torque	Time	×
Velocity	Time	15	Velocity	Time	5
Acceleration	Frequency	10	Acceleration	Frequency	×
Position Error	Frequency	10	Position Error	Frequency	×
Torque	Frequency	5	Torque	Frequency	×
Velocity	Frequency	10	Velocity	Frequency	5
$\sum$		110	$\sum$		10

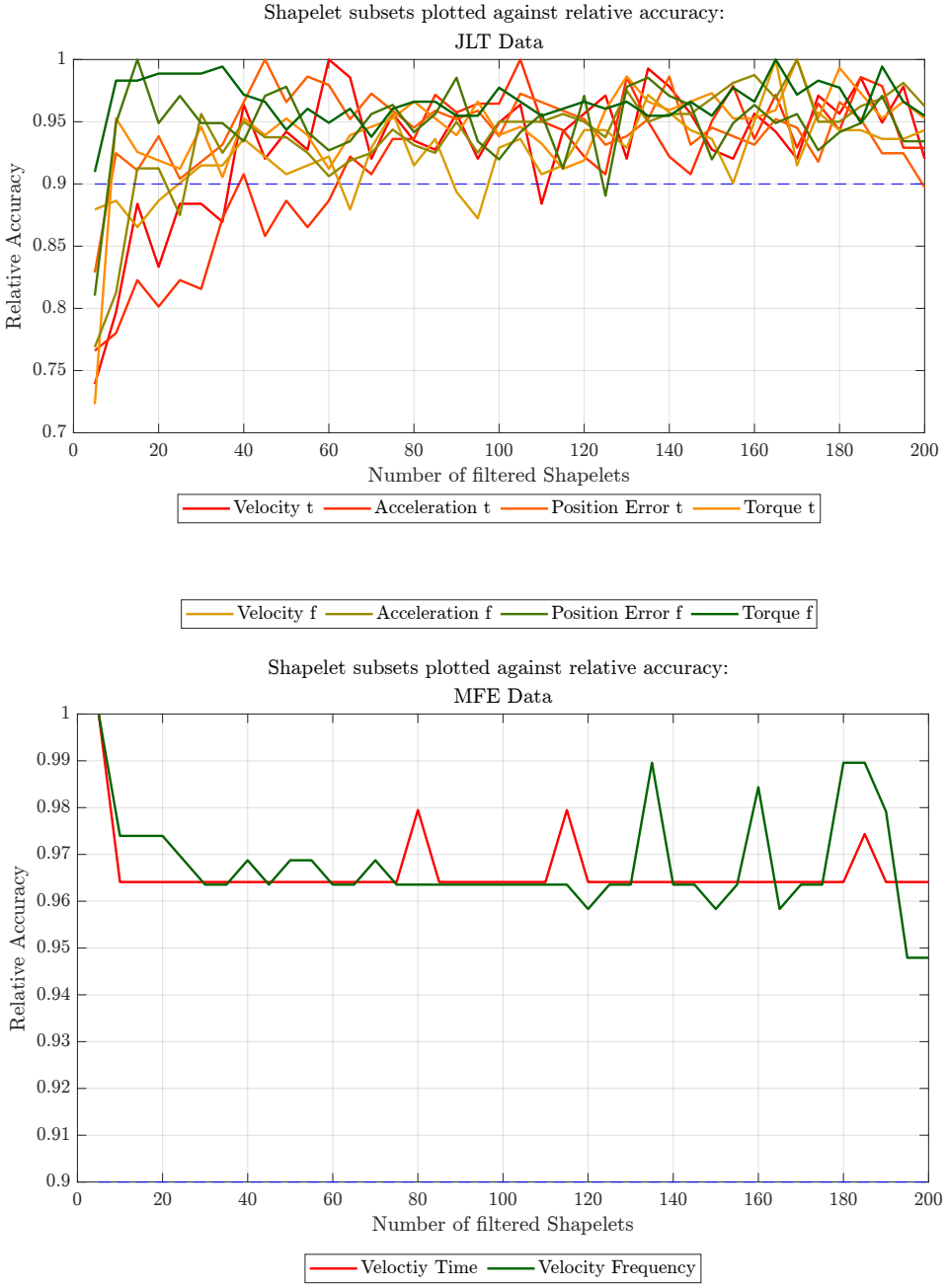


Figure 4.7: Evaluation of different sized shapelet subsets

4.4 Feature Extraction based on Statistical Quantities

To comprehensively evaluate how effectively the given TSC problem is solvable by similarity in shape, features that only capture global properties of time series are also extracted from the provided data. Subsequently, a selected classifier is trained successively with both feature sets

to carry out an expressive comparison. The global features that are employed in this thesis stem from the work of [Feh19]. As described in section 3.3.1, statistical quantities that correlate to the target variable are used as features. From frequency and time-domain, features are extracted from a whole time series. All time-domain quantities employed as feature are listed in Table 4.7. T denotes a time series, m is the respective length, and t_i refers to a data sample of a time series. In frequency-domain, the same quantities as in time-domain are used. Additionally,

Table 4.7: Statistical features in time-domain according to [Feh19]

Feature		Feature	
Mean	$\mu_T = \frac{1}{m}\sum_{i=1}^{m} t_i$	Standard Deviation	$\sigma_T = \frac{1}{m}\sum_{i=1}^{m} (\mu_T - t_i)^2$
Skewness	$T_{\text{skew}} = \frac{\sum_{i=1}^{m}(t_i-\mu_T)^3}{m{\sigma_T}^3}$	Kurtosis	$T_{\text{kurt}} = \frac{\sum_{i=1}^{m}(t_i-\mu_T)^4}{m{\sigma_T}^4}$
Median Mean Ratio	$T_{\text{mmr}} = \frac{\text{median}(T)}{\mu_T}$	Crest-Factor	$T_{\text{crest}} = \frac{\max(\lvert t_i \rvert)}{T_{rms}}$
Clearance-Factor	$T_{\text{clear}} = \frac{\max(\lvert t_i \rvert)}{T_{\text{sra}}}$	Shape-Factor	$T_{\text{shape}} = \frac{T_{\text{rms}}}{T_{\text{arv}}}$
Impulse-Factor	$T_{\text{imp}} = \frac{\max(\lvert t_i \rvert)}{\sqrt{T_{\text{arv}}}}$	Sum of Squares	$T_{\text{ssq}} = \sum_{i=0}^{m} {t_i}^2$

some quantities, which are only used as features in frequency-domain are displayed in Table 4.9. $a(k)$ denotes the amplitude of the k-th discrete frequency of the obtained frequency spectrum.

Table 4.9: Additional statistical features in frequency-domain according to [Feh19]

Feature		Feature	
Energy	$E_i = \sum_{k=0}^{K-1} a(k)^2$	Minimum Frequency	$\text{argmin}(a(k))$
Maximum Frequency	$\text{argmax}(a(k))$		

4.5 Classification Algorithm

In this thesis, all experiments are conducted by training and evaluating a random forest (see section 2.2.3) with the generated features. Since the data used in this thesis is very similar to the data that is investigated in [Feh19], a classification algorithm that delivers good results in [Feh19] is likely to perform well in this thesis, too. Random forests yield the highest accuracies in [Feh19]. Therefore, it is favorable to apply them for an evaluation in terms of accuracy. Moreover, random forests provide another advantage: Bad features do not affect their performance to the same extent as they affect other classifiers' performance. Nevertheless, this is solely an advantage regarding implementation efforts since other classifiers require an upstream feature filter that rejects bad features before they are used for training. This is not the case for random forests.

5 Evaluation

In this chapter, first, all results and evaluations are presented and described. Experiments were conducted to examine accuracy, data requirement, and general suitability of the shapelet approach to the considered TSC problem. After all results are expounded, a subsequent discussion of the results is provided. Especially the comparison between local features, namely shapelets, and global features is described in detail. Eventually, interpretability enhancements of the shapelet method are pointed out.

5.1 Presentation of all Results

5.1.1 Accuracy of Shapelet-Based Classifiction

To comprehensively evaluate the shapelet approach in terms of accuracy, two different experiments are conducted. First, a random forest is trained with the standard hyper-parameter configuration suggested for classification. Here, the minimum node size n_{min} is set to 1, whereas the number of features considered for each split is $m_f = \sqrt{p}$ with p denoting the number of all available features. The number of trees for the random forest is set to $N_{\text{Trees}} = 501$. It should be set sufficiently high, but apart from that, accuracy is not affected significantly by N_{Trees} [HTF09]. With this configuration and the selected shapelet subset of Table 4.5, a random forest is trained and evaluated using 10-fold cross-validation. Therefore, to provide as much training data as possible, train and test data are concatenated to one large data set (899/200 split to 1099 data set) and 10-fold cross-validation (see 2.2.1) is performed by splitting the data ten times using a tenth as test data and the remaining part as training data. Thereon, to examine if individual hyper-parameter adjustments increase accuracy and to reveal the full potential, a SMBO (see section 2.2.4) is applied to the training data. Here, the original train/test splits of both data sets are preserved as described in section 4.2.1. The objective of training two random forests is to provide first an evaluation that is as robust as possible and second to create a classifier as accurate as possible. Evaluating the random forest trained with standard hyper-parameter configuration is very robust since 10-fold cross-validation excludes severe misjudgments of a model's accuracy. Since successful hyper-parameter optimization itself requires sufficient data, the evaluation of a random forest configured accordingly is less robust. The used data for hyper-parameter optimization is worthless to evaluation since results of SMBO may also lead to overfitting. Therefore the results of SMBO can only be evaluated by an

additional evaluation data set. 10-fold cross-validation with this approach is not possible. With these two experiments, two criterions are met: A robust evaluation is provided, and maximum accuracy of the approach is reached, revealing the full potential of shapelets for the given use case.
In Table 5.1 all results and configurations for the JLT and the MFE data set of the performed experiments are summarized. Furthermore, confusion matrixes of all results regarding shapelet accuracy are displayed in Figure 5.1.

Table 5.1: Accuracy of shapelet-based random forests with different hyper-parameter configurations

JLT					MFE				
Configuration	$n_{\min}$	m_f	N_{Trees}	ACC	Configuration	$n_{\min}$	m_f	N_{Trees}	ACC
Default	1	9	501	84.54%	Default	1	9	501	99.82%
SMBO	1	522	501	89.00%	SMBO	19	149	501	99.00%

5.1.2 Data Requirement of Shapelet Discovery

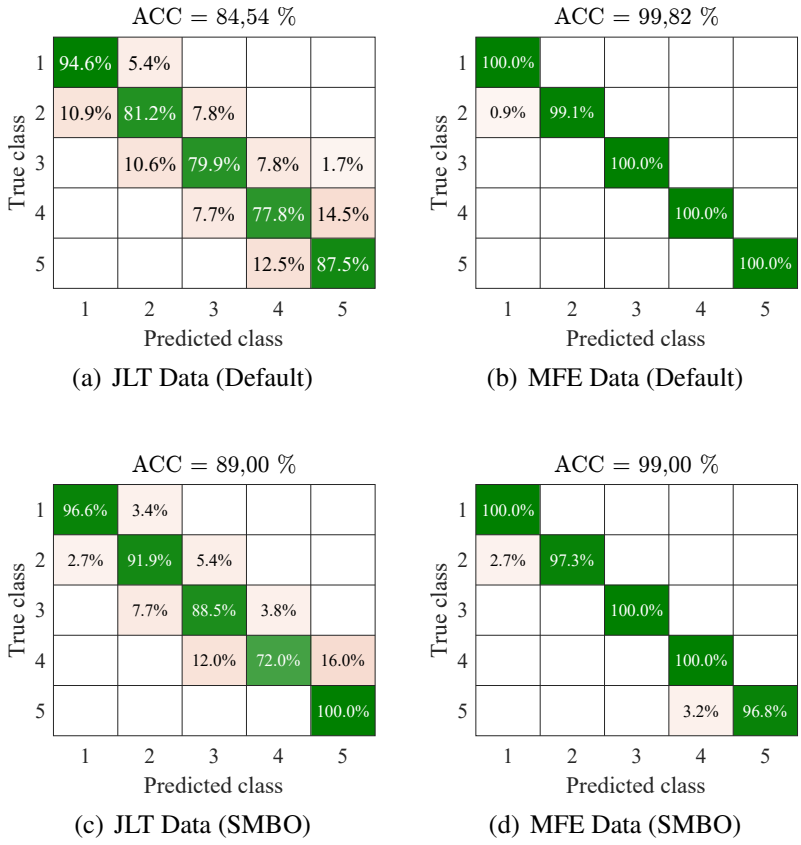
In this section, experiments that aim to determine the data requirement to detect meaningful shapelets are described and evaluated. Therefore, the original data set is shrunk successively. For every further shrunk data set, shapelet discovery according to Algorithm 1 is carried out. The size of each data set that is searched for shapelets is preselected. The range of all investigated data sizes is

$$20 \leq n \leq 500. \tag{5.1}$$

The set of all considered sizes $\mathbf{N}$ contains 25 different sizes, each equidistant to one another with an increment of 20. Thus, the set of all investigated data set sizes is

$$\mathbf{N} = [20, 40, 60, ..., 480, 500]. \tag{5.2}$$

The upper limit of 500 is selected to preserve a feasible run time. This is due to the time complexity of shapelet discovery that is highly dependent on the size of considered data sets. Furthermore, it is expected that after a specific data size, no further improvements regarding shapelet quality are visible since, by definition, good shapelets are found frequently in a data set. Moreover, during all experiments in this regard, a balanced class distribution within each data set is ensured to enable an expressive comparison between each size. That is, for every generated shrunk data set, time series are drawn at random from five different pools that contain only time series of one mutual class. Shapelet search relies on a well-functioning assessment of



(a) JLT Data (Default) (b) MFE Data (Default)

(c) JLT Data (SMBO) (d) MFE Data (SMBO)

Figure 5.1: Confusion matrices of random forests trained by shapelets with default hyper-parameter configuration

shapelet quality. This is only warranted when time series of all classes are available. Avoidance of a balanced class distribution within each shrunk data set would bias the comparison since data sets may not contain time series of all classes. In that case, the shapelet discovery and especially the internal assessment of shapelets is expected to be heavily biased. Both data sets are shrunk according to equation 5.2 and subsequently searched for shapelets according to Algorithm 1. As described, shapelets are assessed during shapelt discovery regarding discriminatory power. However, this evaluation is only based on the available data during shapelet discovery. To evaluate if the detected shapelets of a shrunk data set are generally class-distinctive, the whole originally available data set is transformed by the detected shapelets. After that, a random forest with default hyper-parameter configuration is trained by the obtained shapelets of the shrunk data set and is evaluated using 10-fold cross-validation. The obtained accuracies of this evaluation are expressive of how well shapelets of a specific shrunk data set are generally class-distinctive. In this way, a reliable expression regarding data requirements for shapelet discovery is possible.

The results of these experiments are plotted in Figure 5.2. For both data sets (JLT and MFE), shrunk data sets of the sizes according to equation 5.2 are generated. Subsequently, shapelet discovery is carried out, and shapelets are assessed in terms of general performance.

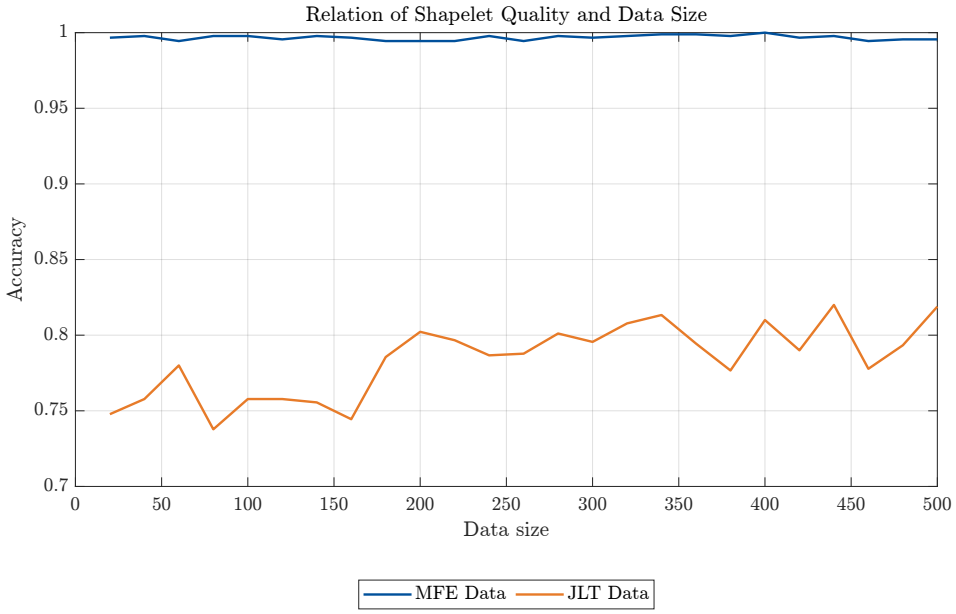


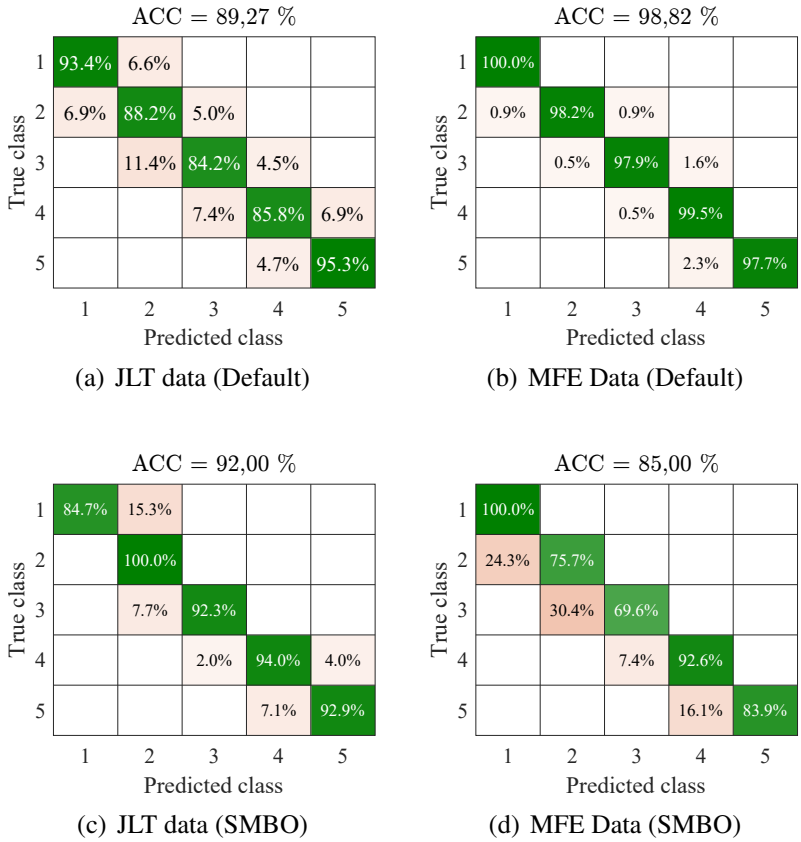
Figure 5.2: Evaluation of shapelet quality from shrunk data sets

5.1.3 Accuracy of Classification Based on Global Features

An evaluation if the class distinction of the tackled TSC problem is rather dependent on local patterns or on global properties of a time series, and to assess which approach is more suitable, a classifier based on global features is trained additionally. Therefore, features, as presented in section 4.4, are utilized for training a random forest. Again, two random forests per data set are generated, differing in their hyper-parameter configuration. To expressively compare the shapelet approach to an approach based on global features, the same two procedures for hyper-parameter configuration are applied to a random forest based on global features. Thus, first, a random forest with default hyper-parameter configuration is trained by global features and evaluated using 10-fold cross-validation. Thereafter, a SMBO to find optimum hyper-parameters is carried out, and a random forest accordingly configured is trained and evaluated on an evaluation data set that is uninvolved in SMBO. The results of this procedure are displayed in Table 5.3.
To provide a more detailed view on the classifier's performance, confusion matrices of both hyper-parameter configurations are displayed in Figure 5.3.

Table 5.3: Accuracy of random forests based on global features with different hyper-parameter configurations

JLT					MFE				
Configuration	$n_{\min}$	m_f	N_{Trees}	ACC	Configuration	$n_{\min}$	m_f	N_{Trees}	ACC
Default	1	12	501	89.27%	Default	1	12	501	98.82%
SMBO	16	288	501	92.00%	SMBO	5	893	501	85.00%



(a) JLT data (Default) (b) MFE Data (Default)

(c) JLT data (SMBO) (d) MFE Data (SMBO)

Figure 5.3: Confusion matrices of random forests trained by global features with default hyper-parameter configuration

5.1.4 Time Complexity of Shapelet Discovery

A striking property of shapelt discovery according to Algorithm 1 is the extensive time complexity. In Table 5.5 the duration of an exemplary run of searching shapelets in velocity data with 500 time series is displayed. To expressively evaluate how time-intensive the shapelet approach is to obtain an accurate classifier, data efficiency of shapelet discovery should be taken into account. Therefore, the time complexity of discovering shapelets considering small

data sets ($n = 20$) is also displayed. Again, velocity time series are considered.

Table 5.5: Time complexity of exemplary runs applied to different data sizes

JLT		MFE	
Data size	Duration	Data size	Duration
$n = 500$	513 876,4 s	$n = 500$	516 110,8 s
$n = 20$	473,7 s	$n = 20$	515,3 s

5.2 Discussion

In this section, the obtained results are discussed an interpreted. The approaches are compared regarding accuracy, data requirement, and time complexity and interpretability enhancements of shapelets are discussed.

5.2.1 Accuracy

The results of the considered shapelet subset for classification seem to be promising. For the JLT data, accuracies of $ACC_{\text{JLT,std}} = 84.54\%$ and $ACC_{\text{JLT,SMBO}} = 89\%$ are obtained. Instead, for the MFE data, the approach delivers accuracies of even $ACC_{\text{MFE,std}} = 99.82$ and $ACC_{\text{MFE,SMBO}} = 99\%$. Despite this, global features still deliver better results. Using global features for classification, accuracies of $ACC_{\text{JLT,std}} = 89.27\%$ and $ACC_{\text{JLT,SMBO}} = 92\%$ for JLT data are observed. Applied to MFE data, accuracies of $ACC_{\text{MFE,std}} = 98.82\%$ and $ACC_{\text{MFE,SMBO}} = 85\%$ are achieved. Both approaches perform very well on the MFE data, whereas global features are slightly superior considering JLT data. It is conspicuous that the resulting accuracies of SMBO deviate a lot compared to the results of standard configuration. In most cases, an accuracy boost is perceived by employing SMBO. Considering the global approach, SMBO instead diminishes accuracy. To assess more robustly if the found configurations of SMBO really increase accuracy, the same train/test split that is used to evaluate the results of SMBO is also applied to train a random forest using standard configuration. It was found that in that case, for all approaches, the standard configuration outperforms the configurations of SMBO. Hence, the random forest configured with standard configuration should be considered as best result of the shapelet approach. Moreover, the comparison between shapelets and global features further indicates that the given TSC problem applied to JLT data is rather solvable by global differences. Local patterns seem to be less meaningful. Regarding MFE data, the results do not indicate which method is more suitable. To further investigate which features are more meaningful, the F-statistic metric is also applied to the extracted global

features. Table 5.7 lists the ten best features of each approach in terms of F-statistic. According to this assessment, global features show a clear superiority over the local features concerning JLT data and therefore acknowledge the previously expressed assumption. Instead, shapelets seem to be slightly superior to global features considering MFE data. Both approaches yield approximately similar best features, but shapelets maintain a comparable high F-statistic in contrast to global features.

Table 5.7: Different features evaluated by F-statistic values

JLT		MFE	
$F_{\text{Shapelets}}$	$F_{\text{GlobalFeatures}}$	$F_{\text{Shapelets}}$	$F_{\text{GlobalFeatures}}$
4.74	21.93	29.35	30.86
3.81	20.85	27.73	30.24
3.22	18.88	26.23	23.50
2.85	17.99	24.38	19.83
2.60	6.17	22.02	19.01
2.57	5.56	20.97	9.86
1.80	4.25	19.41	6.58
1.71	2.98	18.38	6.16
1.68	2.93	17.57	4.21

Another meaningful insight, whether global or local features are more meaningful, would be to consider whole time series as shapelets and assess them with respect to F-statistic. Therefore Algorithm 1 is carried out considering only full lengths of time series. In Table 5.9 the previously obtained shapelets and whole time series considered as shapelets are compared. For each variable of each domain, the best and worst shapelet of the resulting list of Algorithm 1 are compared to the best and worst global shapelets. It is noticeable that in time-domain, local shapelets are clearly superior since they win every comparison. In frequency-domain, however, a whole time series apparently contains more meaningful information than fragments of time series. This is probably due to the property that shapelets capture only phase-independent similarity in shape. In frequency-domain, information is composed not only of mere magnitude but especially of the combination of magnitude and a respective position in the frequency spectrum since the position of a deflection in the frequency spectrum determines the value of a specified frequency. In other words, meaningful information in frequency-domain is particularly found in the position, which is not reliably detectable by shapelets.

5.2.2 Data Requirement and Time Complexity

In Figure 5.2 the quality of detected shapelets with respect to the size of the searched data set is displayed. It is noticeable that the performance of shapelet discovery is fairly invariant to the

data size. No great differences of the obtained accuracies are visible. Merely a slight increase of accuracy is perceptible. Considering JLT data, even with a size of 20 time series, 91.2% of the maximum accuracy is obtained. For MFE data, no significant differences at all between shapelets extracted from small data sets and shapelets extracted from large data sets are visible. However, even though good shapelets are already obtained from small data portions, the large time complexity of shapelet discovery is a clear disadvantage compared to global features. Moreover, shapelets are very dependent on the considered trajectory in the sense that for every new trajectory one might realize, new shapelets must be learned. That is, anytime new features from data must be created and found. Statistical features instead are fixed quantities that only need to be measured from requested data sets. The process of first searching for good features entirely disappears when using statistical quantities.

Table 5.9: Comparison of global shapelets and local shapelets

Variable	Domain	F-Statistic Shapelets				F-Statistic Time Series			
		best	Length	worst	Length	best	Length	worst	Length
Acceleration	Time	1.55	20	0.96	20	0.38	483	0.24	483
Position Error	Time	2.57	370	0.88	370	1.80	483	0.68	483
Torque	Time	1.71	370	0.80	320	1.12	483	0.40	483
Velocity	Time	0.97	350	0.60	170	0.90	483	0.40	483
Acceleration	Frequency	1.80	200	0.98	200	1.87	242	0.73	242
Position Error	Frequency	0.89	50	0.19	10	0.96	242	0.42	242
Torque	Frequency	4.74	200	2.45	200	11.49	242	7.33	242
Velocity	Frequency	1.49	110	0.96	70	0.46	242	0.20	242

5.2.3 Interpretability

To assess interpretability enhancements, the best-found shapelets are projected to the best fitting position of each average time series per class so that shapelets highlight differences among time series of different classes. Since shapelets measure solely similarity in shape, mean and standard deviation of each shapelet are adjusted, so that mean and standard deviation of the respective time series snippet are met. Subsequently, plots of each variable displaying the five average normalized time series per class and their respective best shapelets are created. To investigate interpretability enhancements, a case study of time-domain plots that stem from the MFE data set is presented. It can be seen that shapelets highlight subsequences that apparently determine class membership particularly well. Among all displayed classes, one class is particularly striking. The position of Shapelet 3 and Shapelet 4 in the class "Critically Loose" differs compared to the position they take at all other classes. Shapelet 4 apparently stems from

the class "Critically Tight" since it matches the profile there flawless. It is conspicuous that the global course of Shapelet 4 is particularly steep and that it locally oscillates heavily. The course of the class "Critically Loose" at the respective position Shapelet 4 originally stems from is here neither as steep nor as heavily oscillating. Therefore Shapelet 4 is more similar to another snippet of the class "Critically Loose". One could conclude that the absence of Shapelet 4 in the time series "Critically Loose" at the original position it actually stems from might indicate a loose belt or even belt slippage. A loose belt does not transform power as efficiently as a tight belt. By that, the belt withdraws some power (e.g. due to occurring larger friction during slippage), which results in lower velocities of the drive and therefore, at the particular considered position, in a less steep increase of velocity and fewer fluctuations within the course of the velocity. Shapelet 3 may also indicate this assumption. It represents a descent in velocity, again subject to great fluctuations. Time series of the class "Critically Loose" do not provide a similar enough course at the original position it stems from, which is why Shapelet 3 matches another snippet of class "Critically Loose" more accurately. Instead, the course of class "Critically Loose" at the original position of Shapelet 3 describes a rather smaller descent than the descent that Shapelet 3 captures. Smaller descents, as well as less fluctuations, may also be indicative of power losses due to belt slippages. However, these are all assumptions and should not be considered as proven. Merely, these ideas should illustrate how shapelets could be used to interpret classification results and how they might deliver new insights into the data. More important to mention is the general capability of shapelets to highlight class-distinctive parts of time series and their respective positions, which facilitates further exploration of applied data. In the appendix, plots of projected shapelets to every considered class-average time series of both data sets are provided.

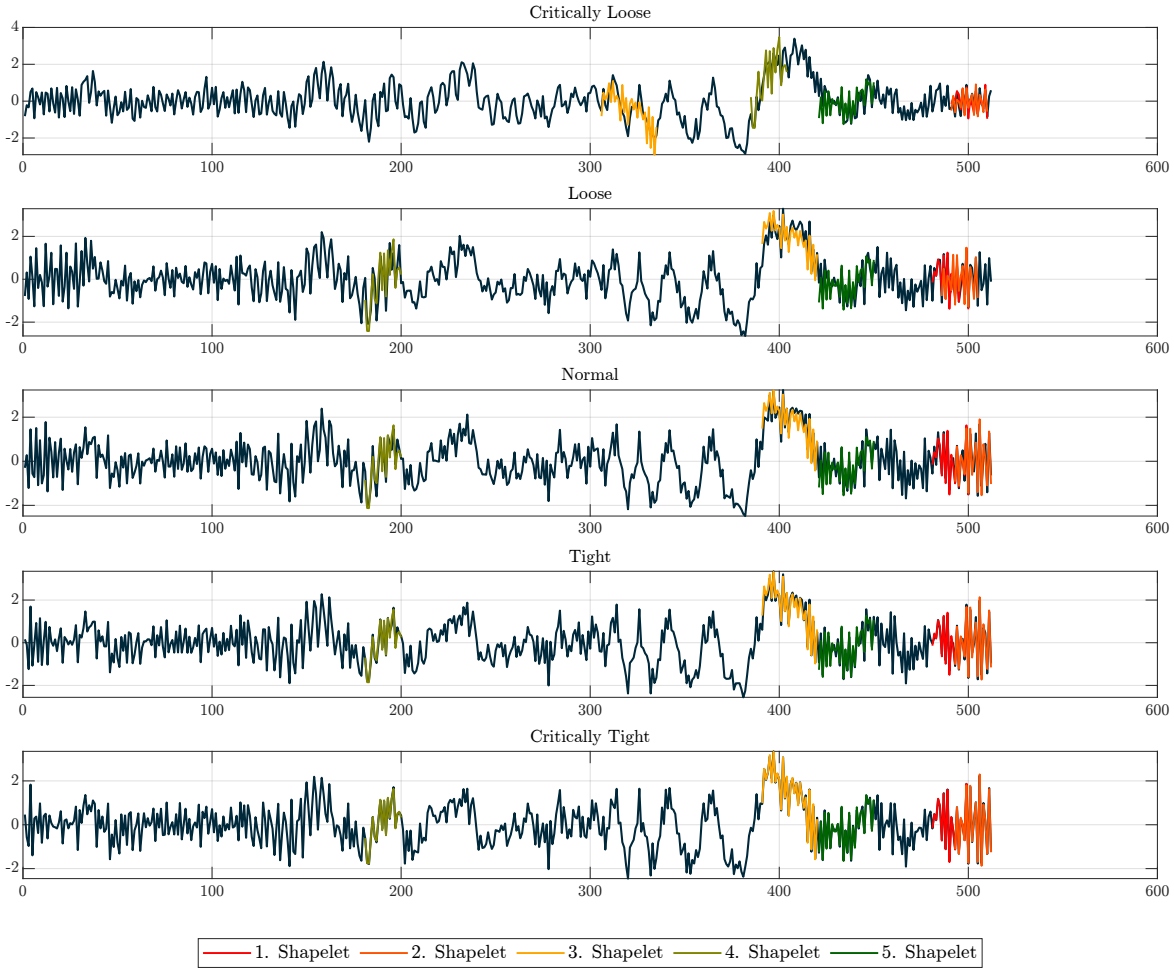


Figure 5.4: Five best shapelets projected to class-average time-domain plots of velocity courses (MFE data)

6 Conclusion and Outlook

In this thesis, a CM of toothed belt drives was successfully implemented. This was achieved by training a ML model to detect the belt tension of an operating belt drive. The model was trained with shapelets. Shapelets are subsequences of time series that are particularly meaningful of class membership. Therefore, an algorithm that searches for shapelets was selected and implemented. The ML model was trained to detect different categories of the belt tension. For this purpose, the measured belt tensions were discretized into five categories: "Critically Loose", "Loose", "Normal", "Tight", and "Critically Tight". Two different data sets were investigated that differ in the specified trajectory. The first data set contains time series of test runs subject to a jerk-limited trajectory (JLT), whereas the test runs of the second data set result from a multi-frequency torque excitation (MFE). The data sets consist of time series with an assigned category. Measurements of the acceleration, position error, torque, and velocity from time and frequency-domain were available. The results show satisfactory accuracies of the implemented CM system built up by shapelets. For the JLT data set, an average accuracy of $ACC_{\mathrm{JLT}} = 84.54\%$ was achieved. Considering MFE data, an average accuracy of even $ACC_{\mathrm{MFE}} = 99.82\%$ was obtained. However, test runs of the JLT data are more comparable to real operation data, which is why ACC_{JLT} should be considered as a benchmark performance. Additionally, the shapelet approach was assessed with respect to data requirements and interpretability. It was found that a successful shapelet discovery is fairly invariant to the data size provided that the data is balanced. Furthermore, possibilities to gain further insights into the data by the shapelet approach were demonstrated. Projecting shapelets to the matching position of related time series enables a more clear visualization of class distinction. Shapelets especially highlight positions that are striking for class membership and furthermore may indicate class membership by mere presence or absence at certain positions of time series. To evaluate if the given TSC problem is rather solvable by global properties of time series or by local properties represented by shapelets, a ML model was trained with global statistical features. The results show that the given time series are rather distinguishable by global properties even though smaller subsequences are almost as meaningful. Furthermore, it is found that especially frequency-domain properties of the given time series determine class membership. A downside of the shapelet approach was the extensive time complexity of shapelet discovery. The run time of the selected algorithm in this thesis is infeasible for large data sets. Investigating data sets of size $n = 500$ was afflicted with run times of $t > 500\,000\,\mathrm{s}$. This is still the case despite the already made modifications in section 4.3.3. Hence, better

shapelets that deliver higher accuracies might still be hidden in the data. Because of that, an application of another algorithm might be worthwhile to examine. [Lu 16] propose a shapelet discovery algorithm that delivers high-quality shapelets by solving an optimization problem. They state, their algorithm is several orders of magnitude faster while achieving even better or comparable classification accuracy. [GWST16] present an algorithm, where an online clustering/pruning technique that discards unpromising shapelet candidates early is implemented, and additionally, incorporates a supervised shapelet selection that filters only shapelets which increase accuracy of an online trained classifier. They state, that their algorithm is three to four orders of magnitude faster and provides better shapelets than benchmark algorithms. Furthermore, to deliver models with both high accuracy and high interpretability, a ML model that is trained with a mix of shapelets and statistical global features might also be worthwhile to examine.

Appendix

Shapelets projected to class-average time series (JLT and MFE data)

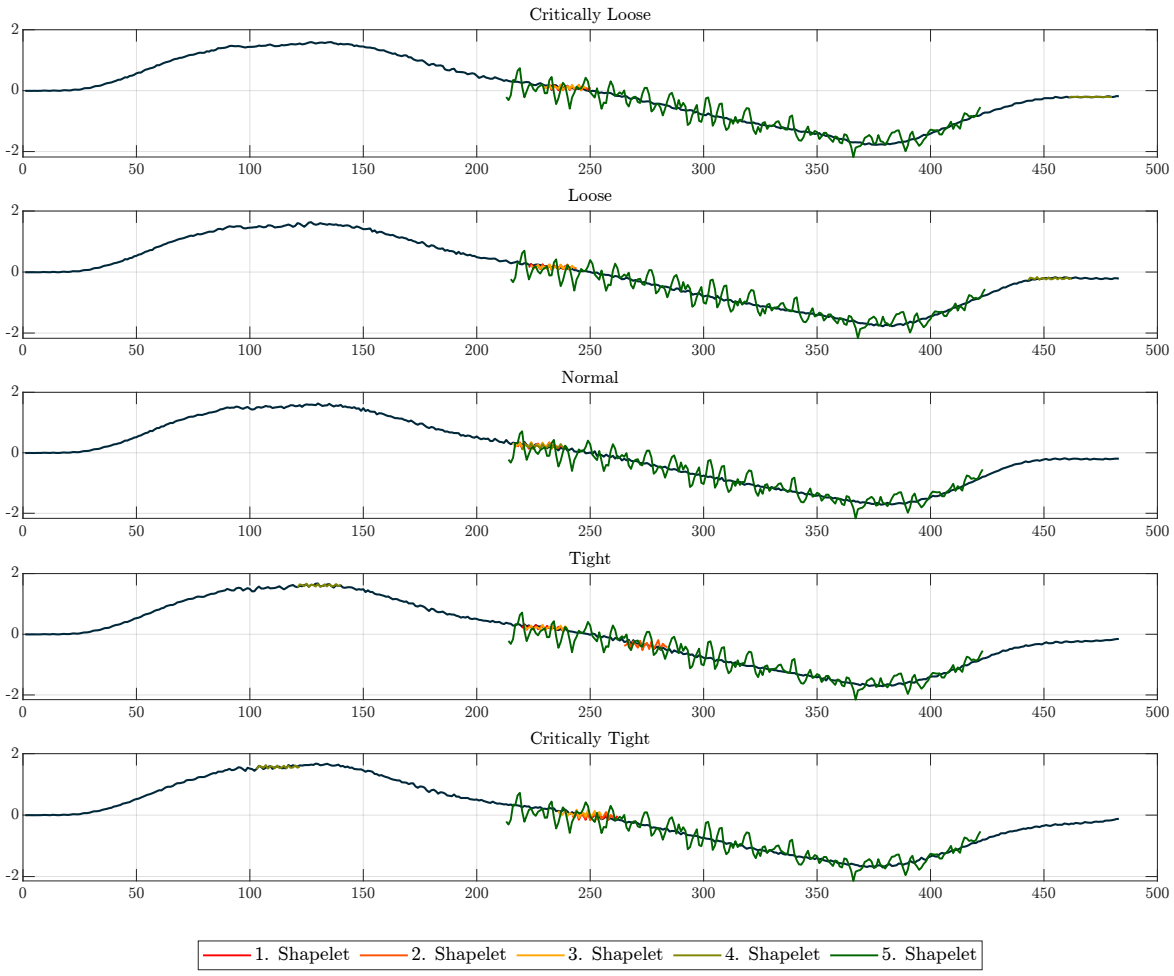


Figure A.1: Five best shapelets projected to class-average time-domain plots of acceleration courses (JLT data)

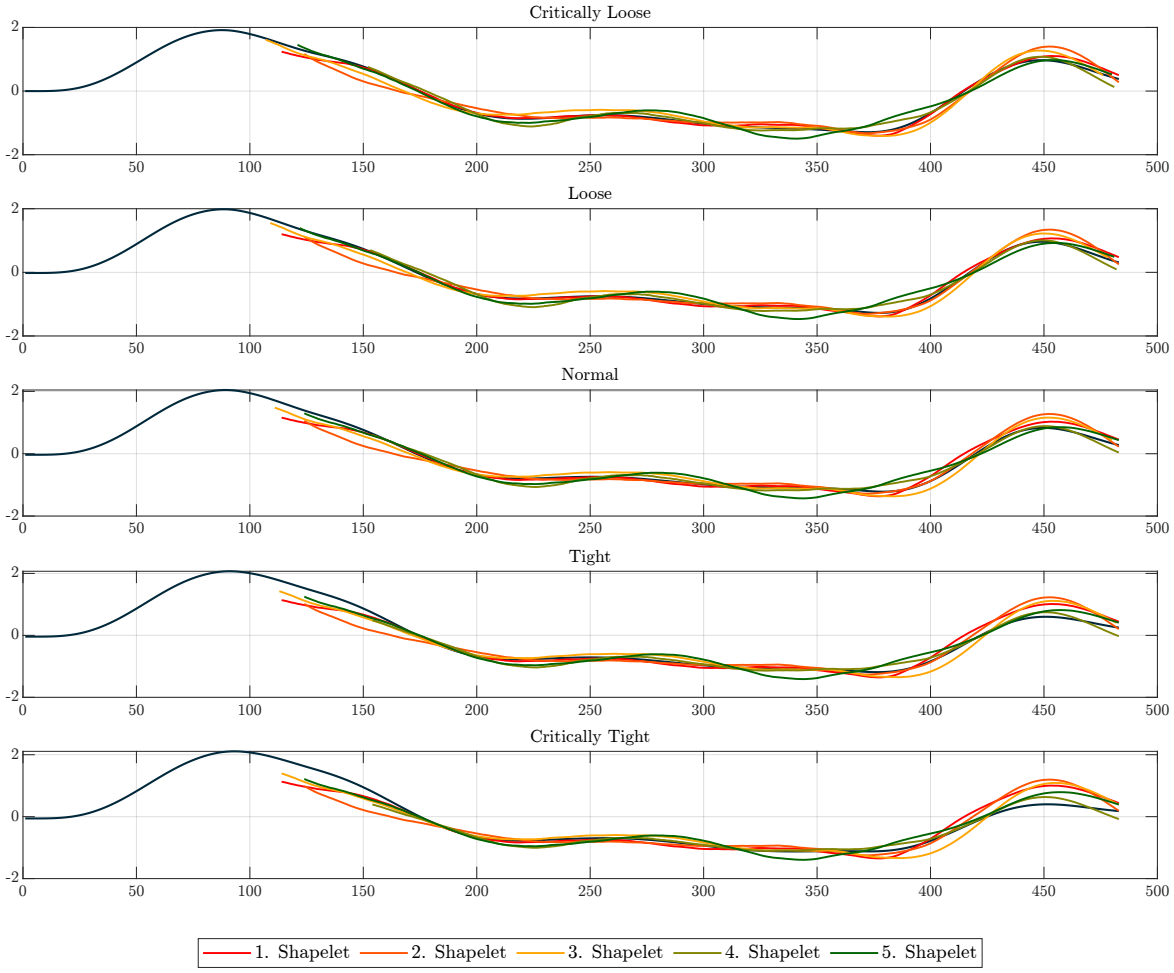


Figure A.2: Five best shapelets projected to class-average time-domain plots of position error courses (JLT data)

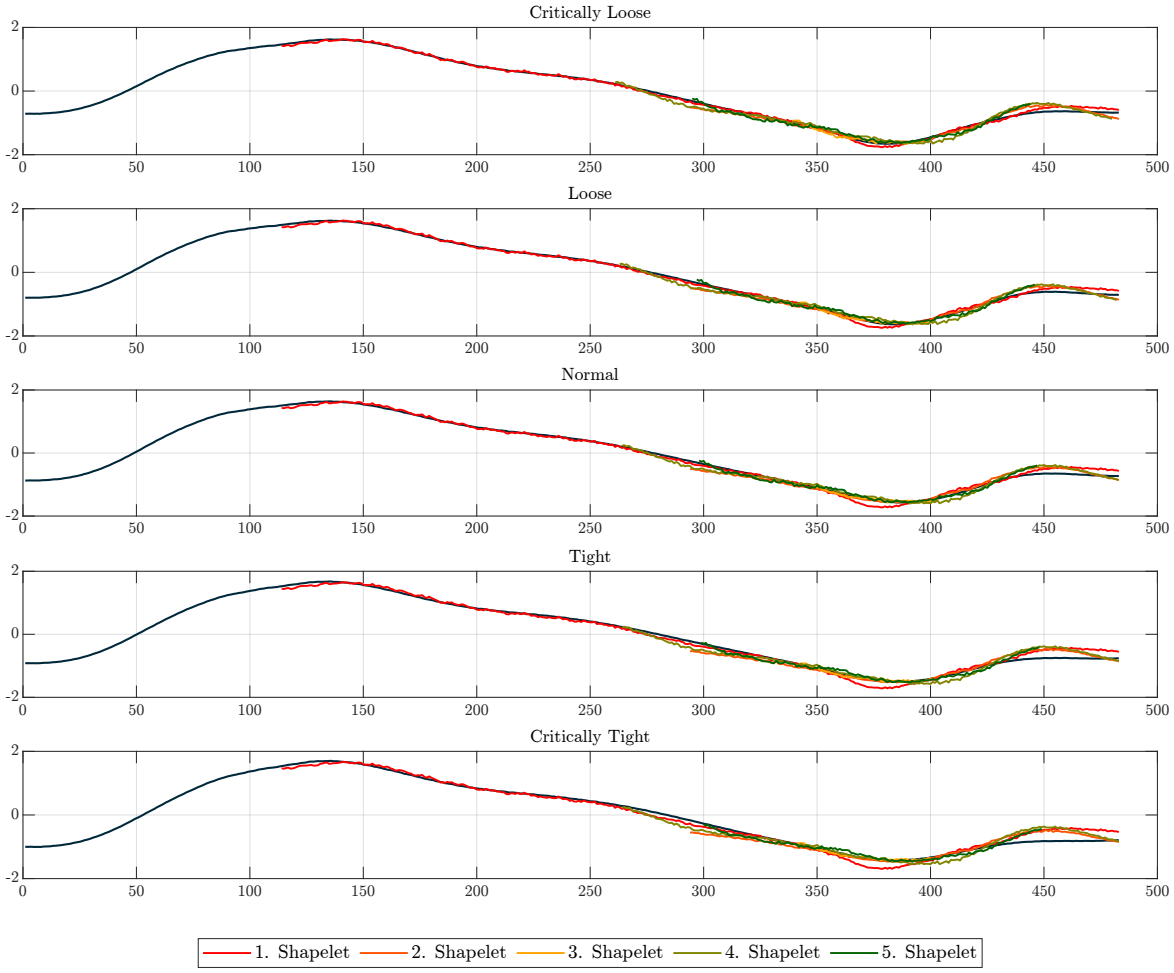


Figure A.3: Five best shapelets projected to class-average time-domain plots of torque courses (JLT data)

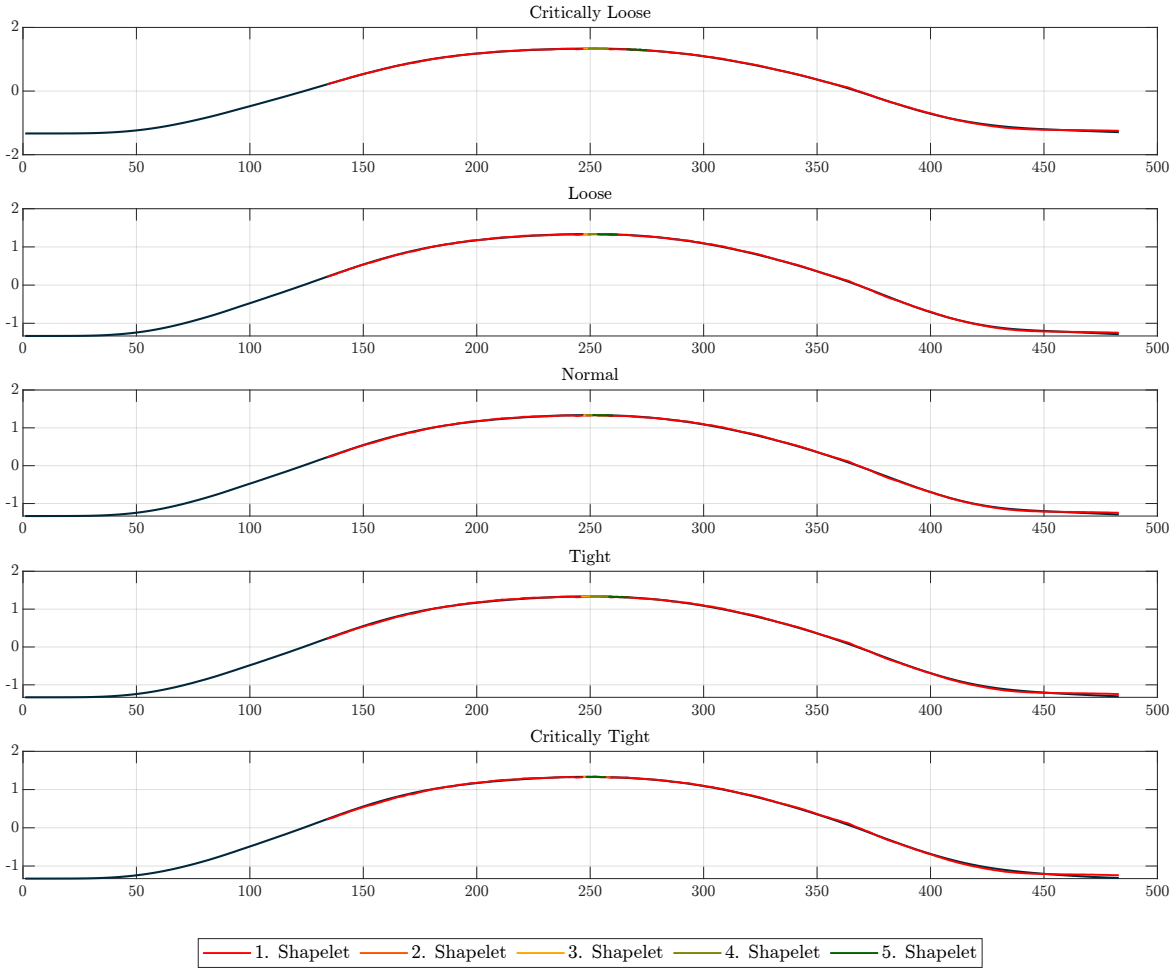


Figure A.4: Five best shapelets projected to class-average time-domain plots of velocity courses (JLT data)

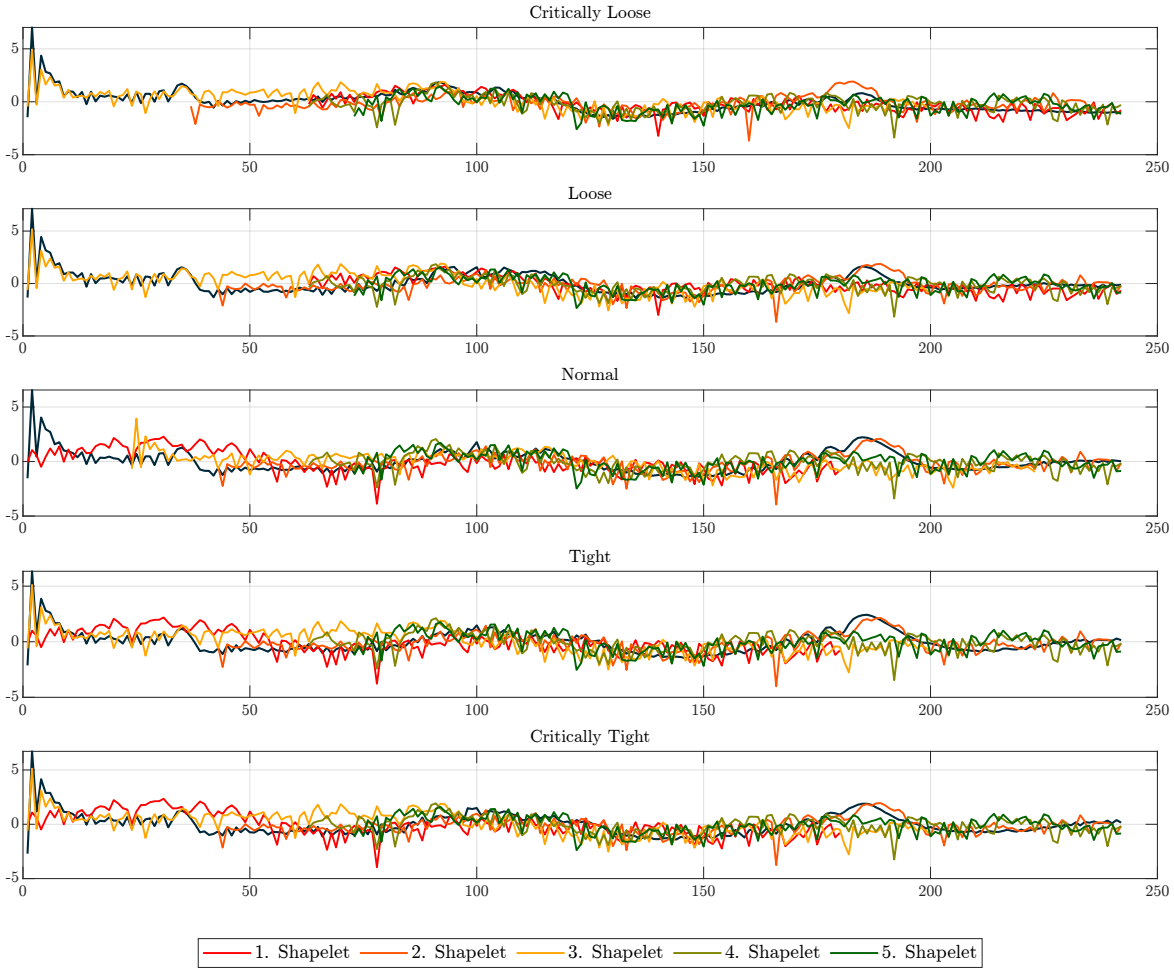


Figure A.5: Five best shapelets projected to class-average frequency-domain plots of acceleration courses (JLT data)

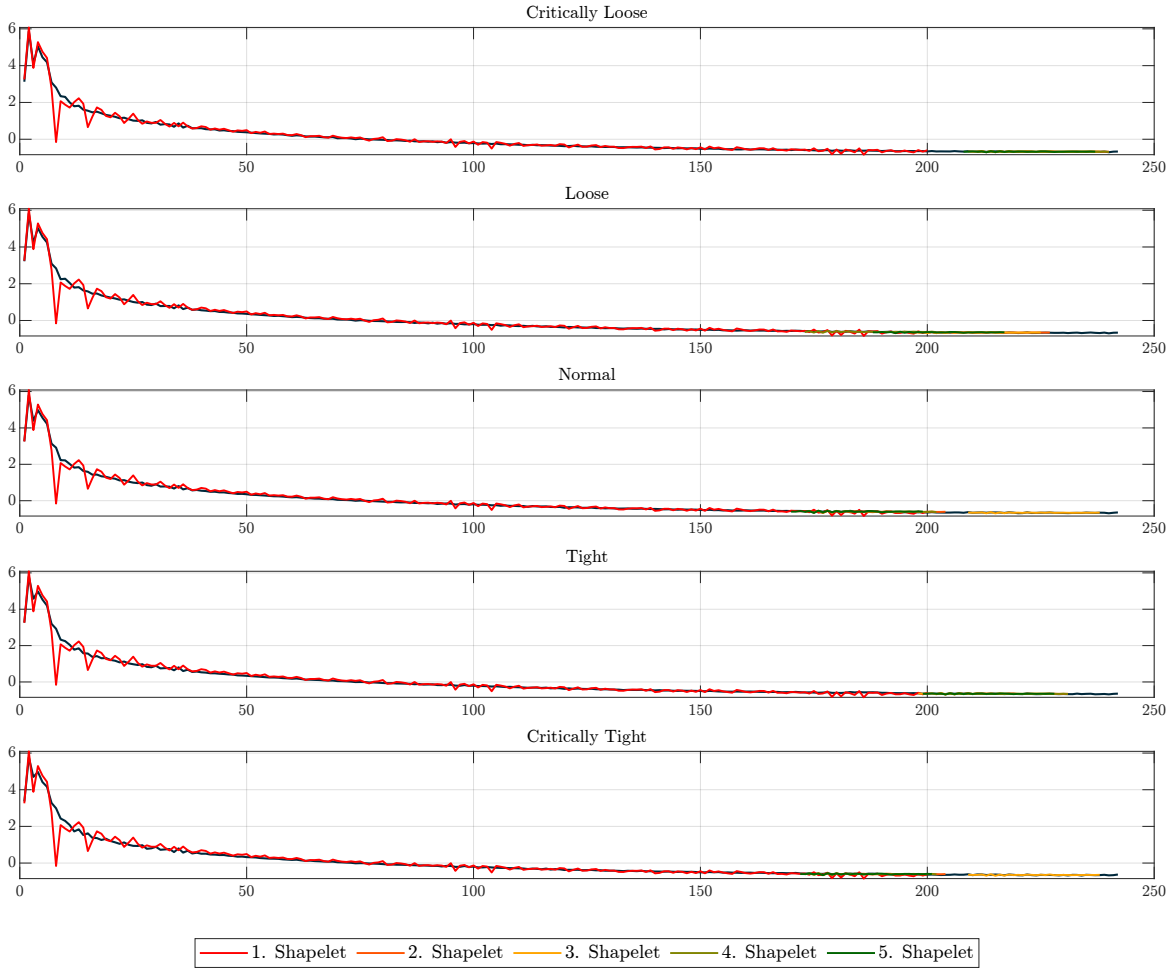


Figure A.6: Five best shapelets projected to class-average frequency-domain plots of position error courses (JLT data)

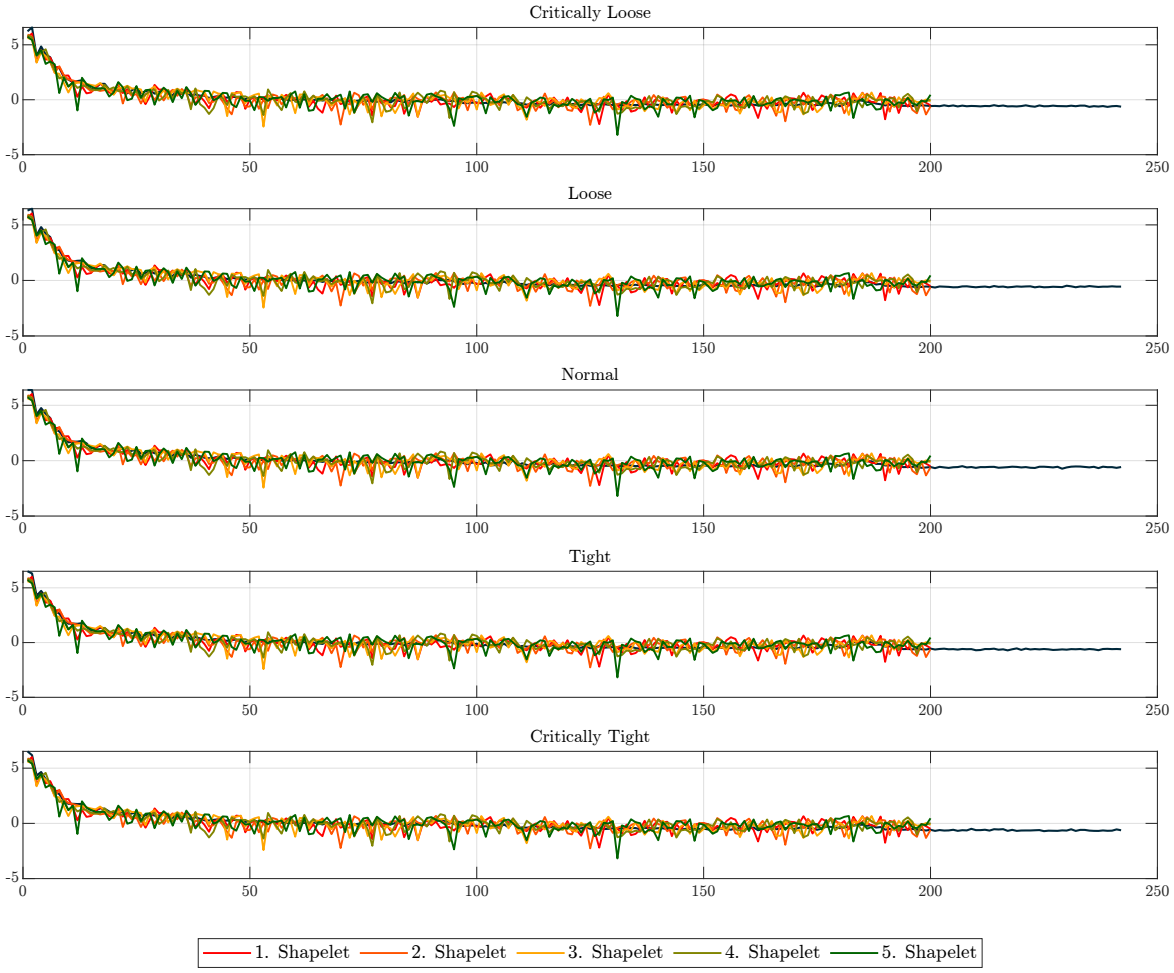


Figure A.7: Five best shapelets projected to class-average frequency-domain plots of torque courses (JLT data)

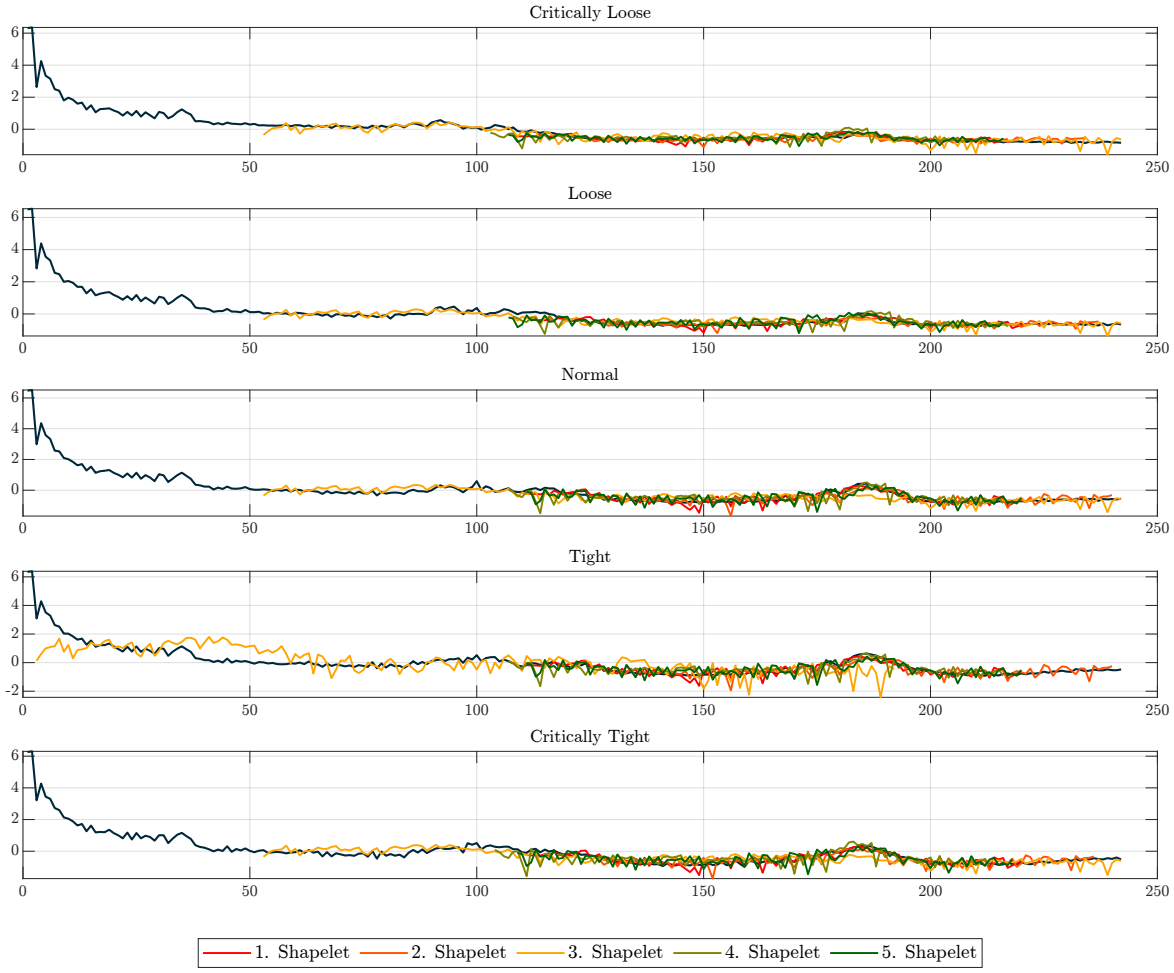


Figure A.8: Five best shapelets projected to class-average frequency-domain plots of velocity courses (JLT data)

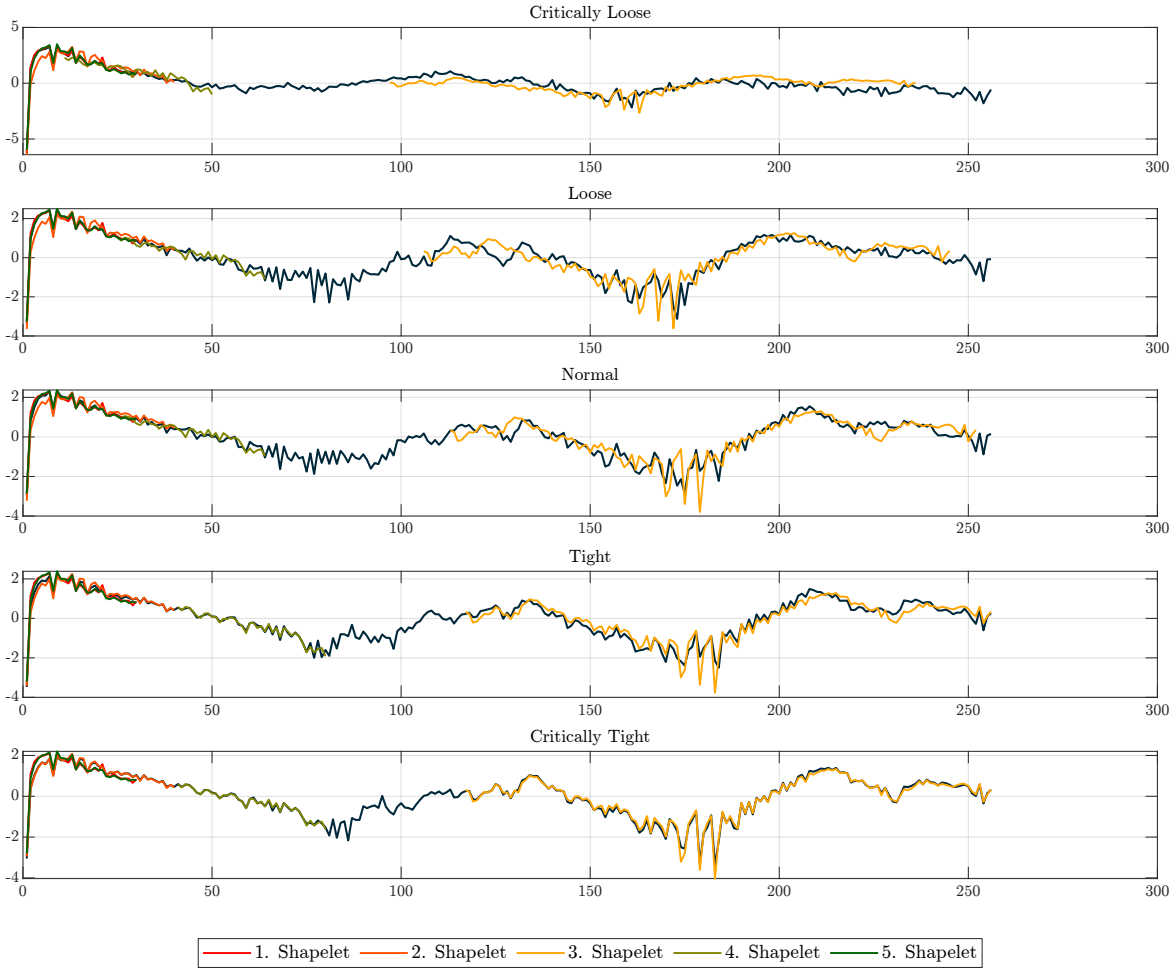


Figure A.9: Five best shapelets projected to class-average frequency-domain plots of velocity courses (MFE data)

Bibliography

[AN19] AHMED, Hosameldin ; NANDI, Asoke K.: *Condition monitoring with vibration signals: Compressive sampling and learning algorithms for rotating machines*. Hoboken, NJ, USA : Wiley/IEEE Press, 2019 https://ieeexplore.ieee.org/servlet/opac?bknumber=8958788. – ISBN 9781119544647

[BB12] BERGSTRA, James ; BENGIO, Yoshua: Random Search for Hyper-Parameter Optimization. In: *Journal of Machine Learning Research* (2012), S. 281–305

[BCF] BROCHU, Eric ; CORA, Vlad M. ; FREITAS, Nando d.: *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. http://arxiv.org/pdf/1012.2599v1

[Bis06] BISHOP, Christopher M.: *Pattern Recognition and Machine Learning*. New York, NY : Springer, 2006 (Computer science). http://swbplus.bsz-bw.de/bsz250316129cov.htm. – ISBN 0–387–31073–8

[Bre96] BREIMAN, Leo: Bagging Predictors. In: *Machine Learning* 24 (1996), Nr. 2, S. 123–140. http://dx.doi.org/10.1007/BF00058655. – DOI 10.1007/BF00058655. – ISSN 0885–6125

[Bre01] BREIMAN, Leo: Random Forests. In: *Machine Learning* 45 (2001), S. 5–32

[Efr79] EFRON, Bradley: Bootstrap Methods: Another Look at the Jackknife. In: *The Annals of Statistics* 7 (1979), Nr. 1, S. 1–26

[Feh19] FEHSENFELD, Moritz: *Untersuchung von Verfahren des maschinellen Lernens zur kontinuierlichen Zustandsüberwachung von Riemenantrieben*. Hannover, Leibniz Universität Hannover, Master thesis, 2019

[FR12] FORTMANN-ROE, Scott: *Understanding the Bias-Variance Tradeoff*. http://scott.fortmann-roe.com/docs/BiasVariance.html. Version: 2012

[FSH15] FEURER, Matthias ; SPRINGENBERG, Jost T. ; HUTTER, Frank: Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015)

[GGB16] GRIMSON, Eric ; GUTTAG, John ; BELL, Ana: *Introduction to Machine Learning*. https://ocw.mit.edu. Version: Fall 2016 (Introduction to Computational Thinking and Data Science)

[GWST16] GRABOCKA, Josif ; WISTUBA, Martin ; SCHMIDT-THIEME, Lars: Fast classification of univariate and multivariate time series through shapelet discovery. In: *Knowledge and Information Systems* 49 (2016), Nr. 2, S. 429–454. http://dx.doi.org/10.1007/s10115-015-0905-9. – DOI 10.1007/s10115–015–0905–9. – ISSN 0219–1377

[HCFC+14] HENAO, Humberto ; CAPOLINO, Gerard-Andre ; FERNANDEZ-CABANAS, Manes ; FILIPPETTI, Fiorenzo ; BRUZZESE, Claudio ; STRANGAS, Elias ; PUSCA, Remus ; ESTIMA, Jorge ; RIERA-GUASP, Martin ; HEDAYATI-KIA, Shahin: Trends in Fault Diagnosis for Electrical Machines: A Review of Diagnostic Techniques. In: *IEEE Industrial Electronics Magazine* 8 (2014), Nr. 2, S. 31–42. http://dx.doi.org/10.1109/MIE.2013.2287651. – DOI 10.1109/MIE.2013.2287651. – ISSN 1932–4529

[HLB+14] HILLS, Jon ; LINES, Jason ; BARANAUSKAS, Edgaras ; MAPP, James ; BAGNALL, Anthony: Classification of time series by shapelet transformation. In: *Data Mining and Knowledge Discovery* 28 (2014), Nr. 4, S. 851–881. http://dx.doi.org/10.1007/s10618-013-0322-1. – DOI 10.1007/s10618–013–0322–1. – ISSN 1384–5810

[HTF09] HASTIE, Trevor ; TIBSHIRANI, Robert ; FRIEDMAN JEROME: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition. New York, NY : Springer New York, 2009. http://dx.doi.org/10.1007/b94608. http://dx.doi.org/10.1007/b94608. – ISBN 978–0–387–84857–0

[HZTM09] HENG, Aiwina ; ZHANG, Sheng ; TAN, Andy C. ; MATHEW, Joseph: Rotating machinery prognostics: State of the art, challenges and opportunities. In: *Mechanical Systems and Signal Processing* 23 (2009), Nr. 3, S. 724–739. http://dx.doi.org/10.1016/j.ymssp.2008.06.009. – DOI 10.1016/j.ymssp.2008.06.009. – ISSN 08883270

[JWHT13] JAMES, Gareth ; WITTEN, Daniela ; HASTIE, Trevor ; TIBSHIRANI, Robert: *An Introduction to Statistical Learning*. Bd. 103. New York, NY : Springer New York, 2013. http://dx.doi.org/10.1007/978-1-4614-7138-7. http://dx.doi.org/10.1007/978-1-4614-7138-7. – ISBN 978–1–4614–7137–0

[KBG+16] KHAZAEE, Meghdad ; BANAKAR, Ahmad ; GHOBADIAN, Barat ; MIRSALIM, Mostafa ; MINAEI, Saeid ; JAFARI, Mohamad ; SHARGHI, Peyman: Fault detection of engine timing belt based on vibration signals using data-mining techniques and a novel data fusion procedure. In: *Structural Health Monitoring* 15 (2016), Nr. 5, S. 583–598. http://dx.doi.org/10.1177/1475921716652582. – DOI 10.1177/1475921716652582. – ISSN 1475–9217

[KM88] KOYAMA, Tomio ; MARSHEK, Kurt M.: Toothed belt drives—Past, present and future. In: *Mechanism and Machine Theory* 23 (1988), Nr. 3, S. 227–241. http://dx.doi.org/10.1016/0094-114X(88)90108-5. – DOI 10.1016/0094–114X(88)90108–5. – ISSN 0094114X

[Koh95] KOHAVI, Ron: A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: *International Joint Conference on Artificial Intelligence (IJCAI)* (1995)

[LDHB12] LINES, Jason ; DAVIS, Luke M. ; HILLS, Jon ; BAGNALL, Anthony: A shapelet transform for time series classification. In: YANG, Qiang (Hrsg.) ; AGARWAL, Deepak (Hrsg.) ; PEI, Jian (Hrsg.): *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*. New York, New York, USA : ACM Press, 2012. – ISBN 9781450314626, S. 289

[LHZ08] LEI, Yaguo ; HE, Zhengjia ; ZI, Yanyang: A new approach to intelligent fault diagnosis of rotating machinery. In: *Expert Systems with Applications* 35 (2008), Nr. 4, S. 1593–1600. http://dx.doi.org/10.1016/j.eswa.2007.08.072. – DOI 10.1016/j.eswa.2007.08.072. – ISSN 09574174

[Lu 16] LU HOU, JAMES T. KWOK, JACEK M. ZURADA: Efficient Learning of Time-series Shapelets. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)* (2016), Nr. 30, S. 1209–1215

[MA07] MCMILLAN, David ; AULT, Graham W.: Quantification of Condition Monitoring Benefit for Offshore Wind Turbines. In: *Wind Engineering* 31 (2007), Nr. 4, S. 267–285. http://dx.doi.org/10.1260/030952407783123060. – DOI 10.1260/030952407783123060. – ISSN 0309–524X

[McK15] MCKINSEY & COMPANY: Industry 4.0 How to navigate digitalization of the manufacturing sector. In: *McKinsey Digital* (2015)

[MKY11] MUEEN, Abdullah ; KEOGH, Eamonn ; YOUNG, Neal: *Logical-Shapelets: An Expressive Primitive for Time Series Classification*. New York, NY : ACM, 2011. – ISBN 9781450308137

[MTMZ12] MEDJAHER, Kamal ; TOBON-MEJIA, Diego A. ; ZERHOUNI, Noureddine: Remaining Useful Life Estimation of Critical Components With Application to Bearings. In: *IEEE Transactions on Reliability* 61 (2012), Nr. 2, S. 292–302. http://dx.doi.org/10.1109/TR.2012.2194175. – DOI 10.1109/TR.2012.2194175. – ISSN 0018–9529

[OK20] ORTMAIER, Tobias ; KNÖCHELMANN, Elias: *Skript zur Vorlesung Robotik II: vollständige Version inkl. Tafelanschriebe*. Hannover, 2020. – vollständige Version inkl. Tafelanschriebe

[PLM96] PACK KAELBLING, Leslie ; LITTMAN, Michael L. ; MOORE, Andrew W.: Reinforcement Learning: A survey. In: *Journal of Artificial Intelligence Research* 4 (1996), S. 237–285

[PO12] PERNEDER, Raimund ; OSBORNE, Ian: *Handbook Timing Belts*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. http://dx.doi.org/10.1007/978-3-642-17755-2. http://dx.doi.org/10.1007/978-3-642-17755-2. – ISBN 978–3–642–17754–5

[RK13] RAKTHANMANON, Thanawin ; KEOGH, Eamonn: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: GHOSH, Joydeep (Hrsg.) ; OBRADOVIC, Zoran (Hrsg.) ; DY, Jennifer (Hrsg.) ; ZHOU, Zhi-Hua (Hrsg.) ; KAMATH, Chandrika (Hrsg.) ; PARTHASARATHY, Srinivasan (Hrsg.): *Proceedings of the 2013 SIAM International Conference on Data Mining*. Philadelphia, PA : Society for Industrial and Applied Mathematics, 2013. – ISBN 978–1–61197–262–7, S. 668–676

[TLB20] TEIXEIRA, Humberto N. ; LOPES, Isabel ; BRAGA, Ana C.: Condition-Based Maintenance Implementation: a Literature Review. In: *Procedia Manufacturing* 51 (2020), S. 228–235. http://dx.doi.org/10.1016/j.promfg.2020.10.033. – DOI 10.1016/j.promfg.2020.10.033. – ISSN 23519789

[TS12] T., Shajina ; SIVAKUMAR, P. B.: Human Gait Recognition and Classification Using Time Series Shapelets. In: *2012 International Conference on Advances in*

Computing and Communications, IEEE, 2012. – ISBN 978–1–4673–1911–9, S. 31–34

[VKS13] VERMA, Nishchal K. ; KHATRAVATH, Sreevidya ; SALOUR, Al: Cost Benefit Analysis for condition based maintenance. In: *2013 IEEE Conference on Prognostics and Health Management (PHM)*, IEEE, 2013. – ISBN 978–1–4673–5723–4, S. 1–6

[WCW07] WANG, Ling ; CHU, Jian ; WU, Jun: Selection of optimum maintenance strategies based on a fuzzy analytic hierarchy process. In: *International Journal of Production Economics* 107 (2007), Nr. 1, S. 151–163. http://dx.doi.org/10.1016/j.ijpe.2006.08.005. – DOI 10.1016/j.ijpe.2006.08.005. – ISSN 09255273

[XPK10] XING, Zhengzheng ; PEI, Jian ; KEOGH, Eamonn: A brief survey on sequence classification. In: *ACM SIGKDD Explorations Newsletter* 12 (2010), Nr. 1, S. 40–48. http://dx.doi.org/10.1145/1882471.1882478. – DOI 10.1145/1882471.1882478. – ISSN 1931–0145

[YK11] YE, Lexiang ; KEOGH, Eamonn: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. In: *Data Mining and Knowledge Discovery* 22 (2011), Nr. 1-2, S. 149–182. http://dx.doi.org/10.1007/s10618-010-0179-5. – DOI 10.1007/s10618–010–0179–5. – ISSN 1384–5810

[YLGK15] YIN, Shen ; LI, Xianwei ; GAO, Huijun ; KAYNAK, Okyay: Data-Based Techniques Focused on Modern Industry: An Overview. In: *IEEE Transactions on Industrial Electronics* 62 (2015), Nr. 1, S. 657–667. http://dx.doi.org/10.1109/TIE.2014.2308133. – DOI 10.1109/TIE.2014.2308133. – ISSN 0278–0046

[ZYW19] ZHANG, Weiting ; YANG, Dong ; WANG, Hongchao: Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey. In: *IEEE Systems Journal* 13 (2019), Nr. 3, S. 2213–2227. http://dx.doi.org/10.1109/JSYST.2019.2905565. – DOI 10.1109/JSYST.2019.2905565. – ISSN 1932–8184