Universiteit Leiden

Opleiding Informatica

ETA: A machine learning oriented platform for high-dimensional time series analysis

Name:	Lars Hopman
Date:	29/08/2016
1st supervisor:	Shengfa Miao
2nd supervisor:	Thomas Bäck

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ETA: A machine learning oriented platform for high-dimensional time series analysis

L.K. Hopman (S1289152)

August 26, 2016

Abstract

Time series data can be found in a variety of fields such as finance, signal processing, weather forecasting and more. A time series analysis pipeline typically covers a wide variety of tasks such as feature extraction and model generation. To support the full range of tasks we propose an *Environment for Time series Analysis (ETA)*. The platform provides a platform for analyzing time series data aiding both academia and industry. This implies that the functionalities associated with time series analysis are supported inside the platform. Some basic modules for machine learning, time series representation methods, preprocessing and feature extraction are delivered to provide a working platform since the first release. ETA enables the users to expand the application by creating modules on their own. As ETA is written in Python, machine learning packages such as *scikit-learn* can be utilized to gain access to a great amount of existing algorithms. Our application can be used to analyze time series data sets more efficiently.

Contents

List of Figures

Chapter 1

Introduction

Time series data has a wide variety of applications, they can be found in finance [63], signal processing [54], bio informatics [2] and many other scientific and non-scientific fields. Time series can be analyzed to gain more insight in the domain or to create models for predicting values. Time series analysis covers a wide spectrum of tasks. Example tasks are feature extraction, preprocessing and model generation. A tool that is extendible and customizable and would offer a valuable contribution. All these functionality combined offers a platform which can be used to research time series data sets more effectively.
Machine learning is used in more and more fields. With many machine learning algorithms, the applications are diverse. Tasks such as classification, clustering and regression can be utilized to analyze (large) streams of data. When combining automatic learning and data measured at a certain interval, an interesting event of smaller scale can be seen. By combining the knowledge of different scales, better understanding is achieved and interesting new applications can be conceived.
Although many time-series related software exists (shown in section 2.4) they are all focused on a single task in the data mining process. To our knowledge no platform covering all tasks related to time series data mining exist. In this thesis we propose a Python-based [68] framework called *Environment for Time series Analysis* (ETA) for analyzing time series data. ETA offers some basic modules for profiling, plotting, preprocessing, feature extraction and machine learning. All these modules are build with Python libraries such as *Pandas* [42], *SciPy* [29], *Numpy* [10], *Scikit-learn* [52] and *Matplotlib* [26] By applying a modular approach the application can easily be expanded in the future. Future modules can also easily incorporate these or other existing Python libraries.
This thesis will start with a brief introduction in time series data. In the second chapter, some background and definitions are provided. The first section will cover time series specific subjects such as data representation, similarity measurements and indexing methods. This will give an intuition in current time series. The second part will provide a background in machine learning. The different tasks such as regression, classification and clustering are are shown. Also,

machine learning categories as supervised learning and unsupervised learning are discussed. Finally, an overview of existing platforms and their drawbacks is reviewed to show the novelty of this work. The functionalities implemented by these applications are discussed. A comparison with our platform is made to show the differences.
After showing the related work on time series analysis software. A more in depth review is about the implementation of ETA is provided. This includes the architecture and the several types of modules available inside ETA. In the final section illustrations are used to show how a typical machine learning workflow can be applied inside ETA.
Apart from this thesis a tutorial on how to add modules can be found in Appendix A. The documentation of the platform can be found in Appendix B. This tutorial will elaborate on the naming conventions and the knowledge necessary to write a module. After this tutorial the user should be able to implement his own module inside ETA.
ETA is open-source. Open-source software can be used and expanded by every user. To provide easy distribution, the source is available online. Hosting of the The most recent version can be found at Bitbucket[1]. We encourage users to use or expand this platform.

[1]https://bitbucket.org/bachelorproject_timeseries/eta

Chapter 2

Preliminaries and related work

In this chapter the fundamentals of time series are discussed and an overview is given on machine learning. Our platform focuses on time series analysis and machine learning model generation. For the user to work with our platform, a background in machine learning and time series is necessary.

2.1 Time-series data

Time series data is ubiquitous. Many fields make use of time series data. Together with the increase of data in general, time series data is growing more and more. New sources of time series can be found in social media and signal processing. Time series data can be defined as information measured at a certain interval (definition 1). Each t_n is a measurement and should be a real number.

Definition 1. A time-series T is an ordered sequence of m real-valued variables

$$T = (t_1, ..., t_m), t_i \in \mathbb{R}$$

Many applications utilize only a part of the time series. Patel [50] gives us a more formal definition on subsequences of time series data (definition 2).

Definition 2. Given a time series T of length m, a subsequence C of T is a sampling of length $n < m$ of contiguous position from T, that is, $C = t_p, \ldots, t_{p+n-1}$ for $1 \leq p \leq m – n + 1$.

To create an easy understanding of time series data, visualisations can be used. The most simple visualisation is the two dimensional plot. An example of a two dimensional plot containing time series data can be found in (Figure 2.1). More about visualizations of time series can be found in the Bachelor thesis of Joost Martens. He elaborates on the visualization part in his thesis.

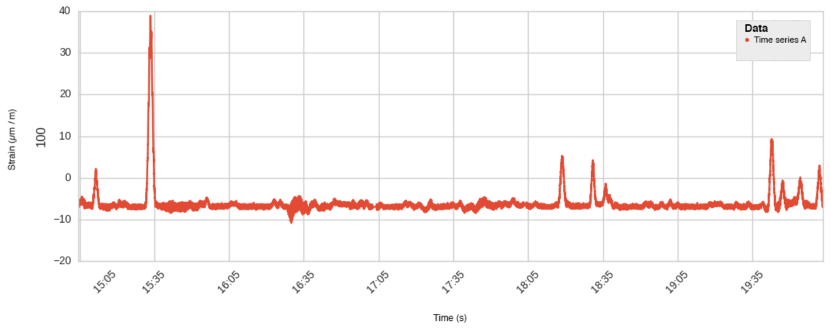


Figure 2.1: An example of a line graph containing time series data.

Many interesting directions arise when working with time series, examples include dimensionality reduction, similarity measurement and data representation. To provide a framework for describing these directions, Esling [15] defines three major categories for working with time series data. Data representation, similarity measurement and indexing method are typical aspects for time series specific data mining tasks. By combining time series specific knowledge with more generic machine learning knowledge to create a machine learning pipeline oriented specifically at time series.

2.1.1 Data representation

As time series are getting bigger and bigger, dimensionality reduction becomes an important factor in time series analysis. The aim of dimensionality reduction is to derive the essential characteristics of the data. This data with reduced dimensionality can be used for further processing. By using dimensionality reduction, computation time and storage could be minimised. Several methods for dimensionality reduction of time series are proposed in the literature. One method of dimensionality reduction is segmentation. Keogh [31] shows us an overview of segmentation approaches and their qualities. We can divide two categories for time series representations: representation in the time domain and representations in the frequency domain.

2.1.1.1 Time domain methods

Data representation in the time domain are oriented at time instead of frequency. Popular approaches for representation techniques apply a sliding window for dimensionality reduction. Several algorithms for data representation in the time domain exist but we will only discuss Piecewise Aggregate Approximation (PAA) and Symbolic Aggregate approXimation (SAX) to gain a basic understanding of how time series representation methods could be implemented.

PAA Piecewise Aggregate Approximation [30] (PAA) presented by Keogh et al is a method for representing a time series in the time-domain. The algorithm works by applying a sliding window approach on the given time series. For each window it calculates the mean and uses this value to describe the complete window. This functionality can be seen in Figure 2.2. Dimensionality reduction through PAA is fast to calculate. Additional to normal Euclidean distance (which can be seen in section 2.1.2), the weighted Euclidean distance can be implemented in the algorithm. PAA is a relative simple approach which can be used for many applications and can be used for time series of different lengths. PAA also has disadvantages. Because PAA uses the mean value of the sliding window, the risk of missing important subsequences exist.

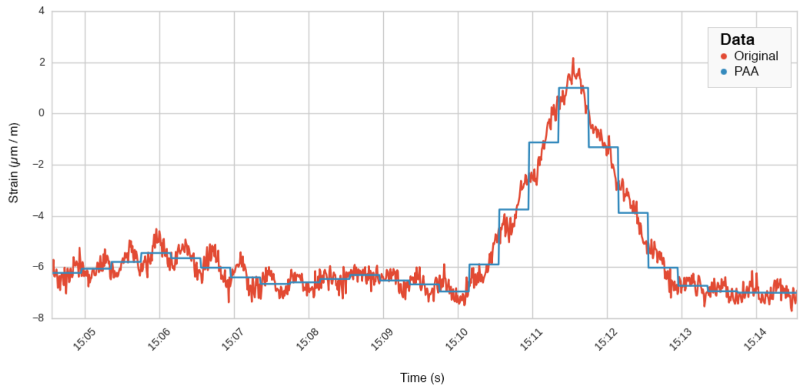


Figure 2.2: A plot showing the original time series and the generated PAA series.

SAX Another time series representation method is Symbolic Aggregate approXimation [38] (SAX). SAX reduces a time series to a string by using a sliding window approach. Before applying the SAX algorithm, normalization is necessary. We first normalize the time series with Z-normalization (shown in 2.3). The normalization of the time series has another advantage: the normalized time series follow a Gaussian distribution. SAX assumes this distribution in the final part of the algorithm. In the next step PAA is applied. Figure 2.3 shows that for each window, the mean of the normalized time series is used for the current window. The alphabet size can be defined by the user. Depending on the alphabet size, breakpoints are calculated. These breakpoints divide the Gaussian curve in n equal-sized areas under the curve. For calculating the distance between strings, a distance measure *MINDIST* is introduced. This distance measure is further introduced in section 2.1.2

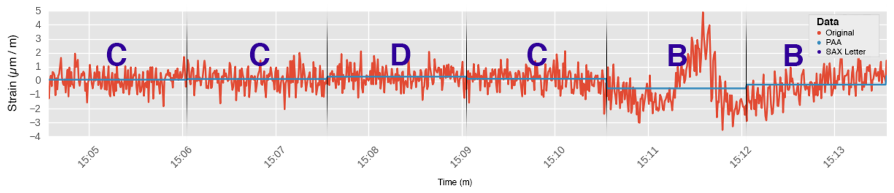


Figure 2.3: Plot showing the original time series, the PAA representation and their corresponding SAX string.

2.1.1.2 Frequency domain methods

Data representation in the frequency domain is oriented at the frequency of the signal. In most cases the original data is in the time domain. When using a frequency domain a conversion is often necessary. When converting a signal from the time domain to the frequency domain, different transformations can be used. Examples are Discrete Fourier Transform (DFT), Wavelet Transform or Laplace Transform. To provide an intuition in the frequency domain DFT is discussed.

DFT Discrete Fourier Transform [25] can be used to transform a signal from the time-domain in the frequency domain. Equation 2.1 shows the formula associated with this transformation. ω is used for the circular frequency. An example of a time series in the time and frequency domain can be seen in Figure 2.4. Several algorithms for applying DFT exist. The most widely used algorithm is the Fast Fourier Transform (FFT). While the original algorithm had a complexity of $\mathcal{O}(n^2)$, the FFT algorithm has a complexity of $\mathcal{O}(n \log n)$ and therefor can be easily used for a high n without high computational cost.

$$f_k = \sum_{n=0}^{N-1} x(n) e^{i\omega nk/N} \tag{2.1}$$

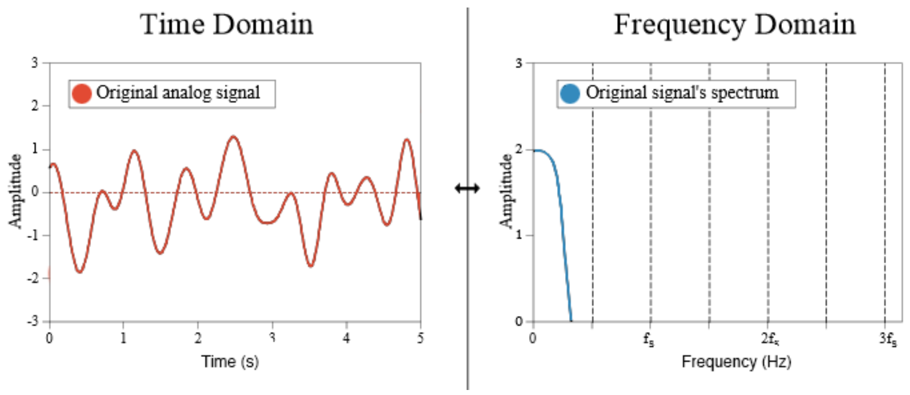


Figure 2.4: An example conversion from a time series in the time domain to the frequency domain from [64].

2.1.2 Similarity measurement

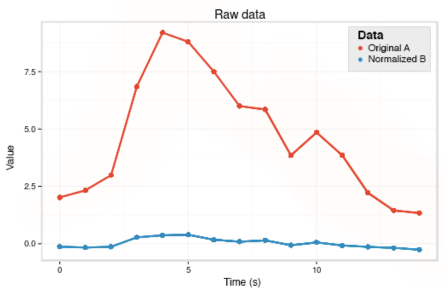
Another important research direction in time series analysis is similarity measurement. Being able to compare time series can provide a valuable asset in analyzing data. Similarity measurement can be used for finding similar subsequences. In this section several methods are discussed for comparing signals and for finding subsequences in streams of time series data.

As Keogh [32] states, comparing offset or amplitude is useless for a non normalized time series. Therefor, normalizing is necessary to provide useful results. Several methods of normalization can be applied. Example of a normalization method is Z-score normalization (shown in equation 2.3). Equation 2.2 shows the calculation of the mean. This equation shows μ as mean and σ as standard deviation. Z-normalization normalizes the values based on these values.

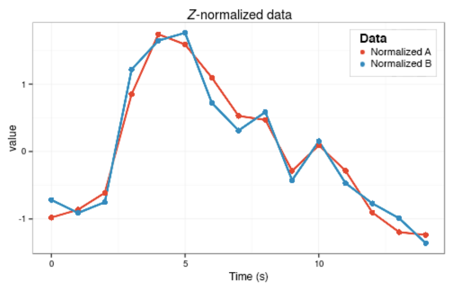
$$\mu = \frac{\sum X}{N} \tag{2.2}$$

$$ZScore = \frac{T - \mu}{\sigma} \tag{2.3}$$

Normalization makes comparing numeric features with different mathematical ranges possible. As can be seen in Figure 2.5a and Figure 2.5b, normalization can help showing similarity of data which first appeared different.



(a) Two raw streams of time series data.



(b) Two normalized streams of time series data.

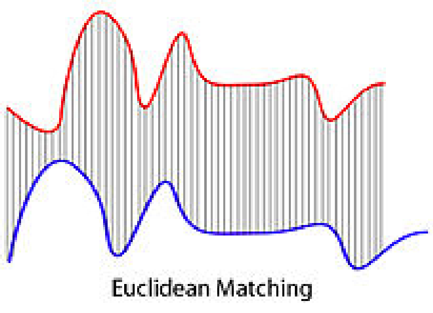
Figure 2.5: The result of normalizing a time series from [28].

Several methods for quantifiying distance between measurements exist. Examples are the relatively simple Euclidean distance, Manhattan distance or Dynamic Time Warping. More methods are discussed in the work of Cha [8]. Again, we provide an intuition of distance measures by looking at two different distance measures: the Euclidean distance and Dynamic Time Warping (DTW) [48]. Sometimes an algorithm specific measure is introduced to quantify the distance between measurements. Non-numeric time series representation method SAX is an example where a new metric (*MINDIST*) is introduced. This metric will also be reviewed.

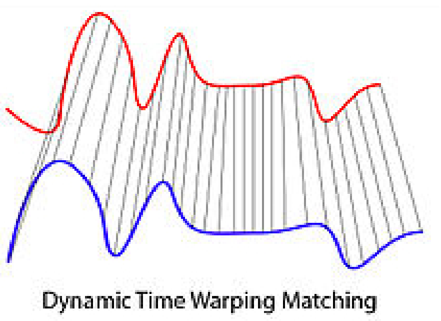
Euclidean distance Euclidean distance is the most widely used distance metric. It relies on the distance between points. The formula to calculate the euclidean distance can be found in equation 2.4. Computation of the euclidean distance is fast and the complexity is $\mathcal{O}(n)$.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2.4}$$

DTW Dynamic Time Warping is a method for measuring distance when the series do not align on the x-axis. A practical example is pronouncing a word at different speed. As can be seen in Figure 2.6a and Figure 2.6b, DTW does not apply a one-to-one mapping as Euclidean matching does. Instead, it tries to make a match based on similarity. DTW origins in speech recognition but the algorithm can be used for many situations where distortion exist on the x-axis.



(a) Mapping of Euclidean distance.



(b) Mapping of DTW.

Figure 2.6: Figure to provide an intuition into the mapping of DTW from [69].

MINDIST As introduced in section 2.1.1.1, SAX utilized a string for representing the time series. As existing numeric-based distance measures such as Euclidean distance can not be applied to string values, a new distance measure is introduced by Lin [38]. SAX works by first generating the PAA representation of the time series. Equation 2.5 is based on the minimum distance between letters in the SAX representation. This equation is derived from the minimum distance between the PAA representation. Often a look-up table is used for calculating $dist(\hat{q}_i, \hat{c}_i)$. $dist(\hat{q}_i, \hat{c}_i)$ gives the minimum distance between two letters in the SAX string. The function accepts two sets of SAX strings.

$$MINDIST(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(dist(\hat{q}_i, \hat{c}_i))^2} \tag{2.5}$$

2.1.3 Motif discovery

An important research direction in time series analysis is motif discovery. Motif discovery deals with detecting patterns and finding similar patterns in a time series. A definition which defines a (wide) scope on motif discovery can be found in definition 3.

Definition 3. A motif is a pair of non-overlapping sequences with very similar shapes in a time series.

To provide more background on motif discovery, we will now go more in depth regarding tasks in motif discovery. As applications range from bioinformatics to speech recognition, different algorithms have been developed. As mentioned before, motif discovery covers a wide variety of tasks including peak detection, finding similar patterns or even finding outliers. Peak detection is an example of pattern detection. After defining a motif, finding similar motifs can also be an interesting application within motif discovery. It aims at finding reoccurring patterns in time series to gain a better insight in the domain.

As Mueen [46] states, motif discovery algorithms have to deal with multiple aspects: the definition of the the motif, domain based pre processing and the algorithm itself. As these aspects are important for creating motif discovery algorithms, they can be used for comparing algorithm types as well.

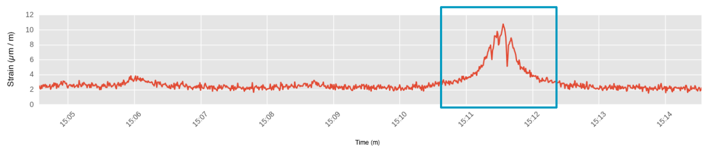
Defining the order of importance of motifs can be an important step as time series are getting longer and longer. With an increasing amount of motifs, the amount of motifs within a time series are increasing as well. By defining a pattern, a scope is created. Different definitions create different subsets of motifs. Mueen gives two examples of definitions and explains the difference. Definition 4 and 5 show how a small difference can lead to a different selection of motifs. Definition 5 focuses on *the most number of repetitions*, which imposes more weight on the repetition part than definition 4 does.

Definition 4. Given a time series and its length, time series motifs are the repeated segments in order of their similarities among the repeated occurrences within an R-ball (data points around a center) .

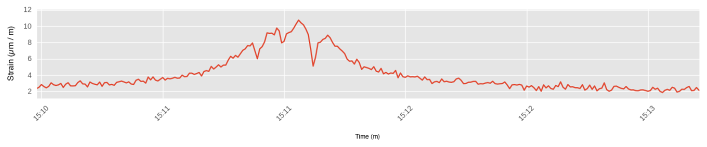
Definition 5. Given a time series and a length, time series motifs are the segments that have the most number of repetitions within an R-ball.

Second, Mueen mentions domain-based preprocessing. Motif discovery is relevant for multiple domains. Preprocessing is necessary to use a motif discovery algorithm. After domain-dependent preprocessing, the algorithm should be able to handle all types of time series data.

Finally, we describe the third aspect of Motif Discovery. The third aspect is about the algorithm and the functionalities. Three aspects for describing the algorithm could be considered: exactness, similarity measure and the representation method used.



(a) A time series.



(b) An interesting subsequence.

Figure 2.7: Motif discovery is about finding and matching subsequences inside time series.

- *Exactness*
  The trade-off between computational cost and exactness of the solution is always a decision in algorithm development. An exact solution is always better than approximation. However, calculating an exact solution is often combined with high computational cost. The creator of the algorithm should weigh these computational cost and the desire for exactness.

- *Representation*
  Section 2.1.1 gives us a brief introduction in data representation methods. Mueen shows in his paper that dimensionality reduction does not impact the results of motif discovery.

- *Similarity measurement*
  The third aspect is related to the distance measure used by the algorithm. Section 2.1.2 gives a background on these similarity measures. The choice of similarity measure affects the result of the algorithm.

2.1.4 Indexing methods

The third category Esling [15] defines for dealing with time series data is the indexing method. As time series are getting larger and larger, space and fast querying becomes more important. An appropriate indexing method should facilitate fast querying with relatively low computational cost. Example indexing methods are Spatial Access Methods (SAMs) [3], shareOur platform is not targeted at implementing different indexing methods. Therefore we only mention this category to provide a complete overview of time series aspects as defined by Esling.

2.2 Feature extraction

Directly applying Machine Learning algorithms to the raw time series data sometimes proves to be ineffective. In most of the cases, feature extraction is applied to tackle this problem. Feature extraction derives new features from the available data while retaining the information included in the original data.

Different types of features can be extracted from time series data sets. Based on scales, features can be divided into three levels (categories). Each extraction category results in a different kind of features.

Data set level This level can be seen as a global approach of feature extraction. The goal of this approach is to select key variables from the original data set. Different approach exist ranging from selecting only the most important features to other methods such as Principal Component Analysis (PCA) [51].

PCA is a dimensionality reduction method used in multivariate statistical analysis. It works by creating a lineair combination to maximize the current variance until the given number of components is met.

Variable level This category can be used to describe the data set on a more general level. An example can be found in extracting peaks and their corresponding height, width and area under curve. Sometimes domain knowledge can be used to create domain specific features. Other, more broad, examples are information about the distribution of the data sets or the mean of the time series. Other relatively simple statistics such as minimum, maximum, frequencies or median can be considered as well. These statistics are used to calculate a value over the whole time series. To calculate this values over subsequences of the signal the sliding window level is introduced.

Sliding window level The sliding window approach is a popular approach for generating time series representation. This method could also be used to generate different type of features. This approach divides the time series in separate windows and calculates values for each window. As said, this approach is also used to create different representation methods such as PAA or SAX.

2.3 Machine learning

Several definitions for machine learning can be found in the literature. Samuel [58] states that machine learning is the field of study that gives computers the ability to learn without being explicitly programmed. Mitchell [45] introduces Machine Learning as the question of how to construct computer programs that automatically improve with experience. A more formal definition from Mitchell can be found in definition 6.

Definition 6. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in X, as measured by P, improves with experience E.

Machine learning algorithms have a wide variety of applications ranging from self-driving cars [22], cancer detection by gene expression monitoring [20], to learnto play the game of Go [61].

2.3.1 Machine learning categories

As machine learning has a wide variety of applications, several tasks can be considered. These tasks such as classification, regression or clustering can be classified in larger machine learning categories: supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning

- *Supervised learning*
  Supervised learning is a machine learning method where, for the training set, the attribute to predict has been provided and can be used to learn. This implies that a labeled data set is supplied. The derived knowledge can be applied to new examples. Examples of supervised learning methods are regression and classification. After model generation, the model should be able to predict new examples.

- *Semi-supervised learning*
  Semi-supervised learning [71] uses both labeled and unlabeled data to perform a machine learning task. Semi-supervised learning can be used when large quantities of unlabeled data are present while having just a small amount of labeled data. Gathering of labeled examples can be expensive or difficult while unlabeled data can be collected easily. Semi-supervised learning uses both supervised and unsupervised learning to generate models.

- *Unsupervised learning*
  Unsupervised learning is the field of study that is researching how to draw knowledge from data without labels. Clustering is an example of an unsupervised learning method. Applications of unsupervised learning methods can be found in bio-informatics for gene clustering and in other field such as pattern discovery.

- *Reinforcement learning*
  Considering a maximizing agent, conditioning can help an agent learning. This process is called reinforcement learning. This process is different from supervised learning in the fact that the correct input and output are never shown. For each mistake a penalty is given and for positive behaviour points are awarded. The algorithm tries to maximize the amount of point to generate the best possible result.

2.3.2 Machine learning tasks

Several data mining tasks associated with unsupervised and supervised learning are discussed in this section. For each task, a definition is given as this will provide us with a scope. After the definition, some basic applications and examples are shown to give a practical background of the task and the goal of the task. Some data mining tasks have smaller subcategories which are used to describe several types of algorithms. If these subcategories exist we discuss them . After that, an intuition is given by explaining a popular algorithm considered part of this learning method. Finally some metrics are reviewd to quantify the quality of the model. These metrics are often task specific and they are discussed as part of the category.

Regression Regression is considered as an important tool for creating machine learning models. Regression is used for predicting continuous values based on given variables. The target value can be considered a lineair combination as can be seen in equation 2.6 where W_p gives the weight for each variable.

$$Y(w, x) = W_0 + W_1 x_1 + ... + W_p X_p \tag{2.6}$$

Regression has a wide variety of applications and can be described as fitting a line to an amount of measurements. Applications are found in many fields such as finance or trend predictions.

Two regression methods can be considered depending on the distribution of the response variable. The fitting of the regression line can be done in a variety of ways. The following, more global methods can be seen:

- *Linear regression*
  Linear regression assumes that the response variable follows a Gaussian distribution. Often, a least-squares fit is applied to generate the model as this is the most simple fitting method of regression.
- *Generalized Linear Model (GLM)*
  A Generalized Linear Model [49] can be used when the response variable does not follow a Normal distribution. For example when the distribution is skewed the GLM can be applied.

Both methods have larger quantity of approaches for fitting the line. Methods such as LASSO regression [65] and LARS [13] can be used when normal methods are not returning the desired result.

After model generation, the model needs to be evaluated to quantify the performance. Metrics can be used to compare the performance of the model. According to Mayer [41], it can be useful to use a variety of measures to determine the quality of the data. A good addition is to visualize the predicted and the observed data as this provide a good first impression about the quality of the model. Below we find a list containing some measures reviewed by Mayer. This is a selection from the wide variety of existing measurements.

- *Root Mean Squared Error (RMSE)*
  The Mean Squared error measures the mean of the square considering the difference between all errors. The y values are used to calculate the difference between the actual value and the predicted value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.7}$$

- *Mean absolute error (MAE)*
  The mean absolute error measures the average error. To calculate the MAE, we again calculate the difference between the predicted value and the actual value. This time we use the absolute value.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| \tag{2.8}$$

- R^2
  R-Squared can be used to measure the performance of the fit depending on the variation of the data. The best possible score is 1.0 which implies that all values are predicted correctly.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} W_i(Y_i - Fi)^2}{\sum_{i=1}^{n} W_i(Y_{av} - Fi)^2} \tag{2.9}$$

Classification Another subset of supervised learning is classification. Classification is used for categorizing examples based on a generated model. Supervised learning implies that labeled data is necessary to train the model. Applications can be found in spam detection, fraud detection and bio informatics (classify proteins). Classification can be done by a variety of algorithms. These algorithms are often called classifiers. Several categories can be defined when dealing with classification problems. The type of preprocessing, algorithm and evaluation metrics are application-dependent.

- *Binary*
  A binary task is the most widely used form of classification. Binary implies two possibilities for the predicted values. An example binary classification task could be to predict if a customer is going to convert in the next year.

- *Multi-class*
  For a multi-class classification task, there is one target value to predict. Possible values x for the target can be $x > 2$ as we are not dealing with a binary classification task. As we are dealing with classification, the target value needs to be categorical. An example of a Multi-class classification task can be found in the *iris* [36] data set where the target value consists of species.
- *Multi-label*
  In contrary to multi-class classification, multi-labelled classification is aimed at predicting different categorical values for multiple target columns. Applications for this classification task can be found in document topic classification. A review of existing methods can be found in the work of Tsoumakas [67].
- *Hierarchical*
  Hierarchical classification classifies the target value in a way that hierarchy is preserved. Applications can be found in text classification and bio informatics. A popular method for the hierarchical structure is the tree. An overview containing current methods discussing advantages and disadvantages can be found in the literature [60].

An example classification approach is the decision tree. We take the decision tree as an example because of the relative simplicity. This will give provide us with an intuition.

A decision tree is a diagram containing the path to different responses. Every branch represents a decision while the leaves represent the possible response. An example of a decision tree can be seen in Figure 2.8. Decision trees can be easily understood and used without much introduction. Two variants of decision trees can be identified. The choice of decision tree type depends on the response variable.

- *Classification trees*
  Classification trees can be used when the response variable is categorical.
- *Regression trees*
  Regression trees are used when the response variable is numerical.

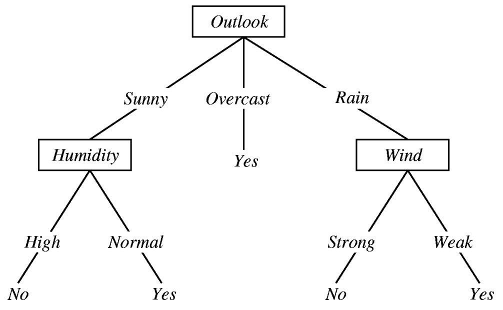


Figure 2.8: An example of a decision tree.

Classification algorithms can be categorized into either decision tree category. Several algorithms are proposed in the literature. The most popular algorithms are described below.

- *ID3*
  ID3 [55] is an algorithm introduced by Quinlan. It creates the split based on information gain. First, the entropy before the split is calculated with equation 2.10. Second, the same formula is used to calculate the entropy after the split. The p value is used to give the distribution of the response variables.

  $$Entropy(C) = \sum_{i=1}^{c} p_i log_2 p_i \tag{2.10}$$

  For each candidate, the information gain (see equation 2.11) is evaluated and the split is created on the largest information gain. This method does not allow for numerical values and therefor can be considered as a classification tree.

  $$Gain(w) = Entropy(C) - Entropy(C|w) \tag{2.11}$$

- *C4.5*
  Regression trees are used when the response variable is numerical. C4.5 is the successor of ID3. Quinlan [56] uses entropy from information theory to generate a decision tree. The main advantage of C4.5 over ID3 is the ability to utilize numeric values for tree generation. C4.5 creates discrete intervals which are used to calculate the entropy.

- *CART*
  CART is introduced by Breiman [7] to create an algorithm to make full use of numerical values. The algorithm finds the best numerical value to create the split on. After tree generation, the full tree is pruned. CART can be classified in the regression trees category as full support of numerical values is supported.

The algorithms mentioned above are popular algorithms and are used for a variety of applications. Again, the goal of this overview is to provide an intuition in machine learning. Although different classification algorithms can be found in the literature, often a combination of algorithms is used to produce better results. Examples of these 'ensemble methods' can be found in Random forests [6], Bagging [5] and Boosting (such as AdaBoost [19]).

Quantifying the quality of the generated model can be done by using metrics. Just as with regression in subsection 2.3.2, task specific metrics exist. Several popular measures for quantifying the robustness of a model are discussed. The discussed measures are a selection, many more measures exist. The discussed metrics are focused on a specific feature of the model (ie. the exactness of predicted positive values). bird-eye view can be achieved by looking at the confusion matrix.

- *Accuracy*
  Accuracy is the most popular and basic performance measure. It works by calculating the percentage of correct predicted values. Equation 2.12 shows us formally how the *true positives* t_p and *true negatives* t_p is divided by all predicted values.

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{2.12}$$

- *Recall*
  Recall calculates the percentage of identified positive labels. It is aimed at indicating how complete the positive prediction is. The formula to calculate the recall can be seen in equation 2.13. A high number of False negatives can be found when the recall value is low.

$$Recall = \frac{t_p}{t_p + f_p} \tag{2.13}$$

- *Precision*
  Precision is aimed at indicating how exact the positive prediction is. As recall is more focused on completeness, precision is focused on exactness.

$$Precision = \frac{t_p}{t_p + f_p} \tag{2.14}$$

- *F1 Score*
  The F1 score combines the values calculated by precision and recall. Therefore it combines the measures of exactness and completeness into a new measure. The combination of these measurements can be seen in equation 2.15.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.15}$$

- *Confusion matrix* The final model quality quantification method we discuss is the confusion matrix. Table 2.1 shows a confusion matrix. On the diagonal we see the *true negatives* and the *true positives*. Our example shows a confusion matrix for a binary classification problem but the matrix can be expanded when applying it to different classification problems.

Table 2.1: Confusion matrix

		Prediction	
		Yes	No
Actual	Yes	T_p	F_n
	No	F_p	T_n

Just like regression performance metrics, these classification performance measures are often combined to create a better understanding of the quality of the model. The measures mentioned above are just an overview of popular performance measures. Other metrics exist and have been analysed in the literature. A more comprehensive overview can be found in the work of Sokolova [62].

Clustering Clustering is a method which can be classified as unsupervised learning. The goal of clustering is to identify structure in an unlabeled data set. Graepel [21] gives us a more formal definition, which can be found in definition 7. Many definitions found in the literature mention a grouping criteria states Estivill-Castro [17]. To provide a more high level intuition, the individual clustering approach should be more focused on this grouping criterion.

Definition 7. Clustering methods aim at partitioning a set X of D data items x_i into N groups C_r such that data items that belong to the same group are more alike than data items that belong to different groups. These groups are called clusters and their number N may be preassigned by the analyst or can be a parameter to be determined by the algorithm. The result of the algorithm is thus an injective mapping $X \to C$ of data items x_i to clusters C_r.

Clustering can be used in a wide variety of situations. The fields of applications are immense, examples include marketing and social sciences.

The literature provides several options for categorizing clustering methods [57]. Han and Kamber [27] propose a division in three categories: density-based methods, model-based clustering and grid-based methods while Farley and Raftery [18] suggest for only two different classes (hierarchical and partitioning). Estivivill-Castro proposed a categorization containing five categories: Partitioning, Hierarchical, Density, Grid, and

Model methods. This categorization gives us clear scope to describe the differences between the categories.

- *Hierarchical*
  In contrast to the other proposed methods, hierarchical clustering creates clusters which are ranked inside a dendogram. Two approaches for creating a cluster hierarchy can be defined.
  - *Agglomerative hierarchical clustering*
    The approach of agglomerative hierarchical clustering starts with each node as a separate cluster. Each step clusters are merged based on a similarity measure.
  - *Divisive hierarchical clustering*
    Divisive hierarchical clustering can be seen as the opposite of agglomerative hierarchical clustering. This approach starts by creating one cluster containing all measurements. The algorithm divides this cluster into separate smaller clusters until final conditions are met.

  Hierarchical clustering has advantages and disadvantages. According to Rokach, it is versatile and can create multiple partitions which represent different levels of similarity. An important disadvantage is the complexity of most algorithms. The algorithms do not scale well with $\mathcal{O}(n^2)$ complexity.
- *Partitioning Methods*
  Partitioning methods are an approach for clustering data. Most algorithms start with a predefined number of clusters. The algorithm assigns measurements from cluster to cluster until the clusters have converged. An example of a partitioning algorithm is K-Means [39] which is used for a wide range of applications.
- *Density-based Methods*
  Density-based clustering is a method based on the sparseness of the data. It assumes measurements in the same cluster have a similar density. An example density-based method is density-based spatial clustering of applications with noise (DBSCAN) [16].
- *Model-based Clustering Methods*
  Model-based clustering works by optimizing the fit between the data and underlying mathematics. Next to finding clusters, the method also tries to create model of each separate cluster. By using this approach it tries to find the underlying characteristics of the cluster. Models can be created by using neural networks or decision trees.
- *Grid-based Methods*
  Grid-based methods apply a computational fast method. It first separates the complete space into several clusters, then it applies the measurements to the corresponding cluster.

As Estivill-Castro states, *"Clustering is in the eye of the beholder"*. The decision for clustering algorithm depends on the data and can be difficult when objective measures do not show significant differences.

Now that we introduced clustering and the several categories of clustering, we can go more in detail on clustering of time series. Liao [35] defines three different approaches for dealing with time series clustering: raw-data-based, feature-based, and model-based. Liao gives a categorized overview and their corresponding applications.

- *Raw-data-based*
  This approach uses the raw data from the time or the frequency domain to cluster the time stream. This is the most straight forward approach.
- *Feature-based*
  The feature-based approach first extracts features before applying the clustering algorithm. A feature extraction example could be to create vectors based on the maximum variance such as Principle Component Analysis.
- *Model-based*
  Model-based time series clustering assumes an underlying logic is represented in the time series. Therefore it tries to generate a model based on the raw time series and applies clustering to the model.

When dealing with a method for unsupervised learning, quantifying the quality of the model is not as straight forward as it is with supervised learning. Sometimes labels are present but most of the times only unlabeled data is available. In the case of unlabeled data, internal metrics from the model need to be used to show the quality of the clusters. Different indices are available in the literature. Examples include the distance between two clusters, the Sum of Squared Error (SSE), Dunn index [12] and the Davies-Bouldin index [9]. While the Dunn index is focused on maximum inter-cluster distance and minimum intra-cluster distance, the Davies-Bouldin index focuses more on the average similarity between clusters. An overview of several performance metrics has been done by Maulik [40]. While indices are available, it remains difficult to quantify the quality of a clustering algorithm without domain knowledge.

2.4 Comparison with other time series tools

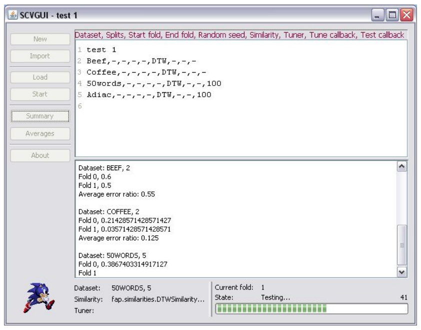
To be able to show the novelty of this work, we first need to look at the tools currently available in the literature. To compare these tools we will especially look at the supplied features and if the system can be easily expanded.

2.4.1 FAP

With FAP, Kurbalija [33] proposes a framework for analyzing and predicting time series data. The framework supports preprocessing, different representations and data mining algorithms. The outline of the framework is discussed in the paper. It takes the same modular approach as our application: the user should be able to easily implement new modules. Unfortunately not all presented features are implemented yet. The Java-based framework have not been update since March 2011. The FAP tool differs from our tool because our framework is based on Python. Next to that, our tool already offers modules for the complete process. Although not all possible modules have been implemented yet, we still offer a tool which can be used for a range of tasks upward of the first release. What's more, the Python language creates freedom for implementing the already widely available modules for machine learning, preprocessing and feature extraction, such as *Scikit-learn* or other modules from the *SciPy* stack. An example of the GUI can be seen in Figure 2.9a.

2.4.2 Weka

Weka [24] is a broadly used tool inside the academic world. From version 3.7.3 and upwards, the tool includes the ability to develop time series forecasting models [1]. The application chooses a machine learning approach by transforming the data in a way that algorithms can process. Afterwards it can apply various types of machine learning algorithms. Weka offers the ability to save and load models for future use. It aims more at creating models instead of offering all tools for the complete data mining pipeline. For example, the tool has no features to apply the different time series representation methods or to extract completely new features. The GUI can be seen in Figure 2.9b.



```
Dataset, Splits, Start fold, End fold, Random seed, Similarity, Tuner, Tune callback, Test callback
test 1
Beef,-,-,-,-,DTW,-,-,-
Coffee,-,-,-,-,DTW,-,-,-
50words,-,-,-,-,DTW,-,-,100
Adiac,-,-,-,-,DTW,-,-,100
```

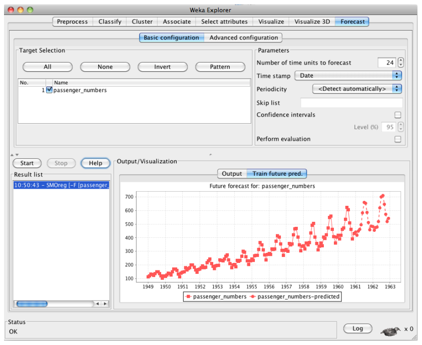
```
Dataset: BEEF, 2
Fold 0, 0.6
Fold 1, 0.5
Average error ratio: 0.55

Dataset: COFFEE, 2
Fold 0, 0.21428571428571427
Fold 1, 0.03571428571428571
Average error ratio: 0.125

Dataset: 50WORDS, 5
Fold 0, 0.3867403314917127
Fold 1
```

```
Dataset:     50WORDS, 5                       Current fold:   1
Similarity:  fap.similarities.DTWSimilarity...  State:          Testing...        41
Tuner:
```

(a) The GUI of the Framework for Analysis and Prediction (FAP).



No.	Name
1 ✓	passenger_numbers

Parameters
Number of time units to forecast: 24
Time stamp: Date
Periodicity: <Detect automatically>
Skip list:
Confidence intervals: ☐
Level (%): 95
Perform evaluation: ☐

10:50:43 - SMOreg [-F [passenger_

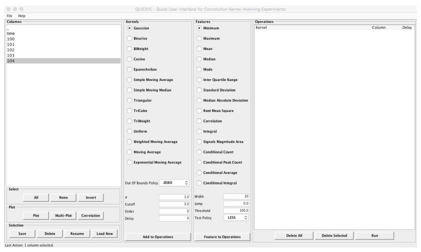
Status
OK

(b) The GUI of the time series analysis part of Weka.

Figure 2.9: GUI of two time series analysis and prediction tools.

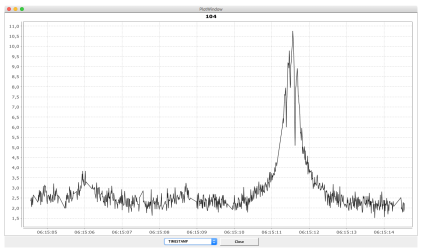
2.4.3 Quickie

The Quick User Interface for Convolution Kernel-Involving Experiments (Quickie) can apply different kernels and feature extraction methods. This Java-based program applies these methods on the loaded data and adds them as new columns. The user can not add new modules to the application due to a lack of modularity. Next to that, the user is not able to apply different representation methods such as SAX or PAA. These methods can be used to reduce the dimensionality of the data. Unfortunately dimensionality reduction can not be achieved inside Quickie because the program only offers the ability to add new features as an additional column. In Figure 2.10a and 2.10b we can see the lay-out of the Quickie application.



Out Of Bounds Policy ZERO
σ 1.0
Cutoff 3.0
Order 0
Delay 0
Width 10
Jump 0.0
Threshold 100.0
Test Policy LESS
Last Action: 1 column selected.

(a) Overview.



(b) Plot.

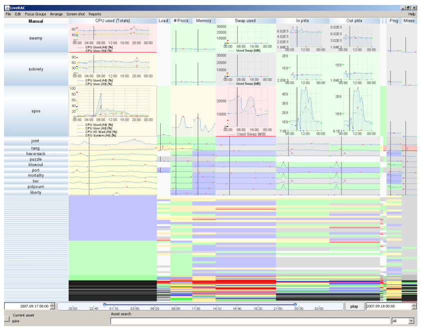
Figure 2.10: Two examples of the GUI of Quickie.

2.4.4 LiveRAC

With LiveRAC [43], McLachlan presents a tool for browsing and correlating time series for system management. The tool aims at providing an overview of the loaded measurements. As can be seen in Figure 2.11a, LiveRAC uses a matrix lay-out for displaying large amount of input devices. It applies a client-server model to divide the database and the visualization application. The visualization part of the application, LiveRAC, must connect to a back end for the data to be able to generate the desired plots. This offers scalability and performance to the application. LiveRAC does not aim to be an application for all the features associated with data mining.

2.4.5 ChronoLenses

ChronoLenses [70] is an application for exploratory analyis of time series data. The application aims at integrating user actions and visual analysis. It can create advanced views and creates the necessary data-transformations on-the-fly. The approach of ChronoLenses is to derive new time series data from the selected original data. This new time series is used to apply different functions and parameters. An interesting feature of ChronoLenses is the visible pipeline which can be used to undo previous steps. The goal ChronoLenses is aimed at visualizing data. Our goal on the other hand is more diverse and comprehensive.

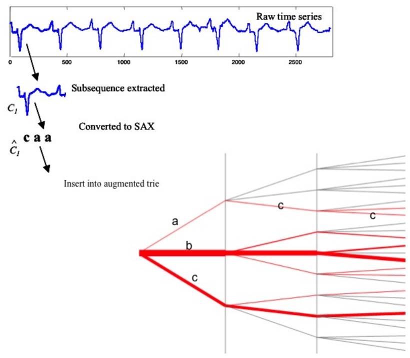

(a) LiveRAC.

(b) ChronoLenses.

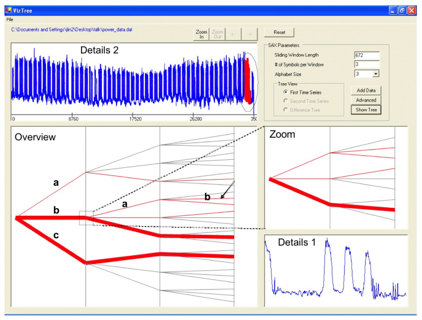
Figure 2.11: Two tools targeted at exploratory analysis of time series data.

2.4.6 VizTree

To be able to monitor the data generated by space vehicles, Lin et al created VizTree [37]. VizTree is a tool developed for mining archival data and monitoring incoming live telemetry. The goal of VizTree is to visualize the results of pattern discovery. It works by displaying augmenting suffix trees based on Symbolic Aggregate Approximation (SAX) as can be seen in Figure 2.12a. The thickness of the lines is generated by the frequency of certain branches. Thick lines represent common patterns, while the thin lines may show anomalies. In Figure 2.12b we see an overview of the application. This application has another goal than our platform does. The functionality is specified for one task (outlier detection) instead of offering a bird-eye view.



(a) The generation of the augmenting suffix tree.



(b) Overview of the application.

Figure 2.12: Viztree, a tool for finding anomalies.

Chapter 3

ETA

As section 2.4 shows, no existing software covers the complete range of functions required to analyze time series. Hence, we propose an *Environment for Time series Analysis* (ETA).

3.1 Architecture

As discussed, ETA uses a modular approach. While the core contains functions for the GUI and handling of the data in the background, the modules apply the transformations. This approach provides a framework suitable for handling all types of data transformations. The core of the program consists of the following classes:

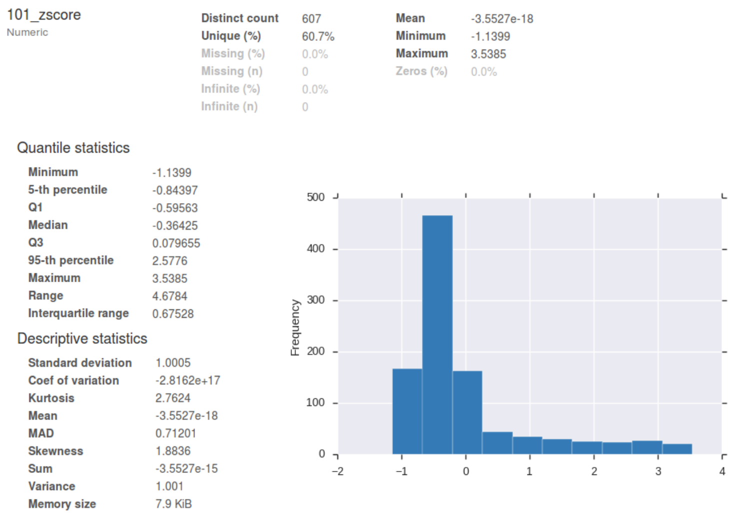
- `guiHandler()`
  This class is designed to handle all functionalities regarding the user interface and the user interaction.
- `dataFramesHandler()`
  This class keeps track of the loaded DataFrames and provide functionalities for modifying these DataFrames.
- `filesHandler()`
  This class is utilized for loading and saving files.

Next to these basic classes, classes exist for communicating with the modules. More information about the design choices can be found in the thesis of Joost Martens. The flow of the data between these classes and the modules will be discussed in the sections regarding the module type.

3.2 Profiling

To gain a better understanding of the data loaded in the platform in an easy way, profiling is implemented in ETA. Profiling is used to create an overview of the data. This can be useful to ascertain missing values, high correlation between variables or odd measurements. ETA offers several options to inspect a loaded data set.

- *Pandas-profiling*
  Pandas-profiling [53] offers a diverse overview for profiling data. It works by generating a web page containing all columns and their meta data. ETA opens the browser automatically after the module is finished with some necessary calculations.



101_zscore
Numeric

Distinct count	607	**Mean**	-3.5527e-18
Unique (%)	60.7%	**Minimum**	-1.1399
Missing (%)	0.0%	**Maximum**	3.5385
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Quantile statistics

Minimum	-1.1399
5-th percentile	-0.84397
Q1	-0.59563
Median	-0.36425
Q3	0.079655
95-th percentile	2.5776
Maximum	3.5385
Range	4.6784
Interquartile range	0.67528

Descriptive statistics

Standard deviation	1.0005
Coef of variation	-2.8162e+17
Kurtosis	2.7624
Mean	-3.5527e-18
MAD	0.71201
Skewness	1.8836
Sum	-3.5527e-15
Variance	1.001
Memory size	7.9 KiB

Figure 3.1: Example of a Pandas-profiling report.

  Unfortunately, Pandas-profiling has its drawbacks. The tool is pretty slow when dealing with larger quantities of data. The used implementation is limited to 10000 rows. If more rows are given to the module, a sample of the complete data set is used. Figure 3.1 shows several measures generated by Pandas-profiling.

- *Print data in terminal*
  Maybe the most basic feature of them all. This functionality prints the loaded data to the terminal.

- *Visualization*
  Altough this functionality can not be found inside the profiling menu, it is a method to gain a basic insight in the data. ETA offers this functionality through the plotting modules discussed by Joost Martens.

3.3 Visualisation

Visualizing data can help to gain an understanding in the data. The proposed framework offers the ability to create modules for visualization purposes. Numerous examples can be considered. Examples include standard line plots or scatters plots. Most of the Figures used in this thesis are created using our platform. This part is developed by Joost Martens and more details about visualizations and their implementation can be read in his thesis.

3.4 Preprocessing

Preprocessing is the step in which the data is prepared for further use. As machine learning algorithms automatically extract knowledge from this data, relevant and accurate data is necessary. Several problems can be considered when preparing data to be processed by a machine learning algorithm. Problems as missing data values, noise inside the data, can affect the results of the machine learning algorithm. A common expression regarding this phenomenon is ¨garbage in, garbage out ¿ which means data has to be correct and complete to provide useful machine learning results. To solve this problem, preprocessing can be applied to the data. After applying the module, the new data is returned. This new data replaces the changes columns inside the currently selected DataFrame.

Some preprocessing modules are implemented inside our environment.

Imputer The imputer takes care of missing values. This module from *scikit-learn* offers a range of parameters. All these parameters such as the placeholder for missing values and the corresponding strategy could be set within the GUI.

3.5 Feature extraction

An approach to handle the increasing flow of data is to use feature extraction. Feature extraction can be used to derive new values from an initial data set.

This section describes how modules are used inside the application. Once the modules are created, little action from the user is required. After selection, the module is loaded and, if present, a window to set parameters for the selected module is shown. After confirmation, the module is executed and a new DataFrame containing the extracted features is returned. If the amount of rows of the new DataFrame differs from the source DataFrame, the user is asked to create a new DataFrame for the extracted features. This functionality can be used to reduce the dimensionality of the data.

ETA is a complex application containing multiple classes. It is necessary to show how the different classes interact. This pipeline can be seen in Figure 3.2.

After confirming the parameters, `extract()` is ran. The data is pulled from the `dataFramesHandler` and, together with the selected parameters, passed to the module. The returned DataFrame is passed and added to the `dataFramesHandler`. At last, the GUI is updated to contain the created DataFrame.

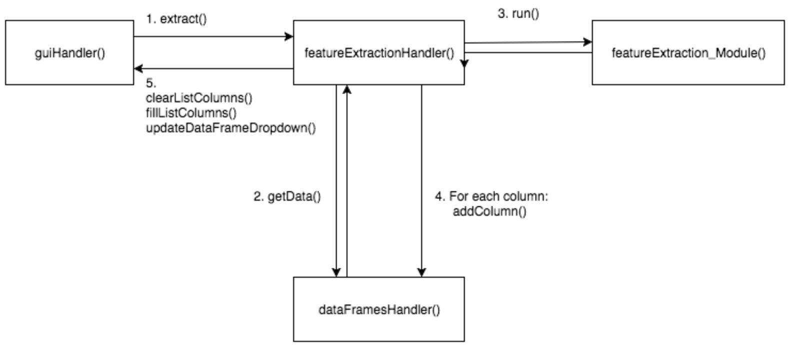


Figure 3.2: Data flow for feature extraction.

In this section several modules are discussed. These modules are able to extract different types of features to show the range of possibilities ETA offer. We show a peak detection algorithm, Principal Component Analysis (PCA) and the Simple Moving Average (SMA).

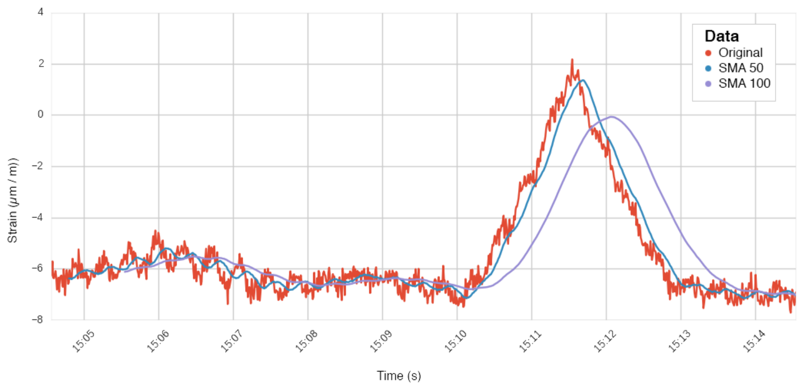


Figure 3.3: Example of a Simple Moving Average.

SMA Simple Moving average or rolling mean has many application and is often used in finance [14, 23, 44] for trend analysis. It smoothens the time series and can be used to detect trends. Our implementation relies on *Pandas* built in function to apply rolling calculations. Different windows can be used by passing values within the GUI. To calculate the moving average, the module uses the formula as can be seen in equation 3.1. An example plot of a SMA with different length of the sliding window can be seen in Figure 3.3. The calculation works by taking the average over the last m values.

$$SMA = \frac{1}{n} \sum_{i=0}^{n-1} p_{m-1} \tag{3.1}$$

Peak Detection A subfield of motif discovery is Peak Detection. The goal of this module is to show that motif discovery modules could be implemented in ETA. Our module creates a new DataFrame based on the peaks of the time series. Several features are considered such as the duration of the peak, the height and the area under the curve.

Our module offers several options. First, the selection of peak detection is necessary. We provide two options for peaks: a method from *SciPy* based on Continuous Wavelet Transformation [11] and a Python implementation of the Matlab Peak Detection algorithm [4]. The selection is done within the GUI.

After the peaks are detected, the features from these peaks can be derived. To calculate these features a baseline needs to be established. This baseline is calculated based on the mean value of the time series. This implies that the time series should not contain an ascending or descending trend. A baseline can be seen in Figure 3.4.

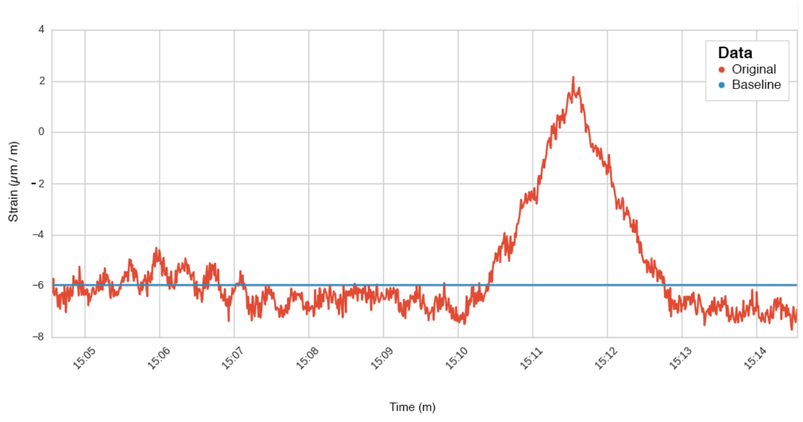


Figure 3.4: A baseline for a time series.

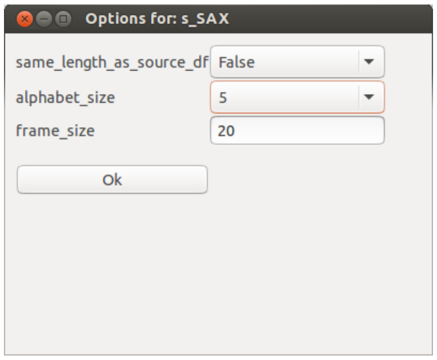
After the baseline has been established. The width of the peak can be calculated. We first check the crossings between the baseline and the time series. These crossings are used as left and right bound of the peak. Finally we utilize the composite trapezoidal rule to calculate the area under the peak. This calculation is done by a function inside *NumPy*.

PCA Principal Component Analysis (PCA) [51] is implemented in ETA to show that feature extraction is possible inside ETA. Our module uses the implementation from *Scikit-learn* which is based on [66]. The user can change the amount of components within the GUI. The generated components are placed inside the current or new DataFrame.

3.6 Representation methods

As discussed in chapter 2, different time series representation methods exist. Representation methods are an important part in time series analysis and are supported in our application. Representation methods can reduce the amount of features necessary to describe the data. Our implementation focuses on creating a framework for deriving new features and translate time series in other representation methods. Different modules are included to show a variety of possibilities. Because of the modularity of ETA, expanding the application with new modules can easily be achieved.

PAA Piecewise Aggregate Approximation as presented by Keoghis a method to change the representation of the time series. The algorithm itself is explained in section 2.1.1.1 of this thesis. The parameters of the algorithm can be adjusted through the GUI of the application. The module offers options to declare the size of the sliding window and an option to make the resulting data the same size as the original DataFrame. The latter functionality can be used to plot the generated values in the same Figure as the original time series. This functionality can be seen in Figure 2.2.



same_length_as_source_df	False
alphabet_size	5
frame_size	20

Figure 3.5: Available options for SAX.

SAX Another time series representation method is Symbolic Aggregate approXimation as presented by Lin [38]. The algorithm is described in section 2.1.1.1. The module features with a number of parameterss such as alphabet size and the size of the sliding window. An illustration of this GUI interface can be found in Figure 3.5. This module uses a look-up table for the letters of the alphabet to provide fast calculations.

3.7 Machine learning

Machine learning can be considered as the last step in the knowledge discovery process. Previous steps such as preprocessing and feature extraction are applied to support machine learning methods by providing useful data. The proposed framework offers a wide variety of functionalities and provides several modules to show the possibilities ETA could offer.

As described in section 2.3, several types of machine learning are proposed in the literature. Both supervised and unsupervised learning modules can be created inside ETA. The workflow for executing these modules depends on the desired data mining task. In supervised learning, a typical workflow could look like this:

1. *Split data in training and test.*
   ETA offers the ability to apply a standard train/test-split on the data. Splitting the data in two separate sets helps to prevent overfitting.

2. *Load model or train model on test set.*
   Training the model needs te be done on the train data set. Several parameters can be adjusted from inside the GUI. Instead of creating a new model, a previously created model can also be loaded.

3. *Use model to predict values on test set.*
   The created model in step 2 can be used to predict values. The prediction is added as a new column to the target DataFrame.

4. *Evaluate the model by using metrics.*
   In supervised learning, one can compare the values that are predicted by the model and the real values. Metrics can be used to quantify the quality of the model. Several metrics can be considered depending on the applied data mining task. Examples of metrics are accuracy, mean squared error or precision. For each module, the returned metrics need to be defined. After applying step 3, a window containing the defined metrics is shown.

5. *Adjust model based on evaluation.*
   Evaluating the metrics can help adjusting the model. It helps to detect overfitting, to check if the model works as expected or to detect flaws. If the model is not performing as desired, step 2 can be redone with different parameters or even with a different algorithm. Evaluation of the model as explained in step 4 needs to be redone as well to measure the outcome of the changed model.

6. *Save model for future use.*
   Finally the created model can be saved as a pickle for future use.

The described typical workflow for classification and regression tasks can be applied inside the proposed framework. The provided options can be seen in Figure 3.6. A short tutorial on how to create a new machine learning module can be found in Appendix A.

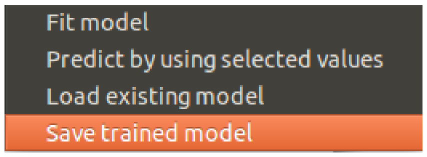


Figure 3.6: Options provided for classification and regression modules.

Unsupervised learning problems require a different approach. As labels are not available, other metrics are required to quantify the quality of the clustering. Another difference is the lacking ability of saving a generated model. Within ETA, the clustering algorithm can be executed. The algorithm parameters can be set within the GUI. After applying clustering, an additional column is added to the DataFrame containing the cluster labels for each measurement.

The most important difference to supervised learning (inside ETA) lies in the fact that no model is generated for future use and no splitting is done on the data.

The flow of the data can be compared with the feature extraction data flow, which can be seen in 3.5. The data is passed along multiple classes. Finally the GUI is updated containing the new DataFrame.

We show two clustering algorithms, one classification algorithm and one algorithm. We are aware that these modules may not cover all the desired functionality. Of course, additional modules can be added by the user.

K-Means K-means is a clustering algorithm based on the Euclidean distance to the cluster centroid. The algorithm randomly places a n number of centroids. It iteratively defines the nearest cluster centroid for each measurements and calculates the new centroid for each cluster based on the measurements currently assigned to that cluster . If the cluster allocation does no longer change, we can say the algorithm has converged. Our implementation relies on Sci-kit learn. Sci-kit learn offers a broad package of clustering algorithms as can be seen in Figure 3.7.

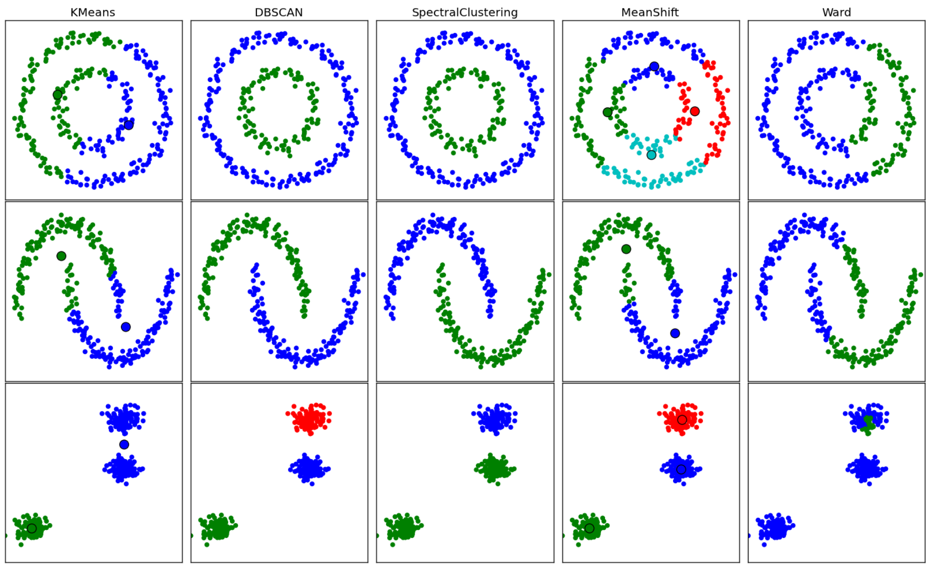


Figure 3.7: An overview of clustering methods inside *scikit-learn* from [59].

Spectral clustering Figure 3.7 shows that Spectral Clustering is another clustering method offered by *scikit-learn*. This method can be used when the structure of the clusters is non-convex. This means that the lines between points inside the cluster does not always lie inside the cluster.

CART CART is a decision tree algorithm which generates the model based on entropy from the information theory. CART supports the use of numerical values. Our implementation uses *scikit-learn* to build the model. The classifier requires no additional parameters. Several metrics are returned to quantify the quality of the generated model. Supported metrics include accuracy, recall, F1 score and the confusion matrix.

LassoLARS regression ETA also implements a regression algorithm to complete the list of machine learning modules. We implemented this algorithm to prove thatregression algorithms can be implemented in ETA. Least Angle Regression (LARS) is a method we can implement to prove this. Again, we use *scikit-learn* because they already offer an implementation of this algorithm. Interesting in this implementation is that cross validation is built in (*LassoCV function in Sci-kit learn*). Little parameters have to be set within the GUI. Only the option to normalize and the desired target are given. Two metrics are returned to show the quality of the model: R^2 and the MSE.

Chapter 4

Conclusion and future work

Literature study shows that no platform offers all time series relevant functionalities. Research shows that most tools focus on only a few specific functionalities.

To provide a solution to this problem, we propose ETA (Environment for Time series Analysis). ETA is a platform which aims to provide a bigger picture when analyzing time series. Our tool adopts a modular approach. Several modules are provided since the first release. Modules for preprocessing, feature extraction, dimensionality reduction and machine learning can be easily created. Our framework is delivered with some standard modules. Example modules include Principal Component Analysis, a Peak Detection algorithm and machine learning algorithms such as CART, K-means or regression. Different time series representation methods such as SAX and PAA are also implemented and can help reducing the dimensionality of the data. With the use of Python, widely used packages such as *scikit-learn* and *Pandas* can be easily utilized inside the application. The platform is offering the complete pipeline from file loading till machine learning model generation.

Our platform fills the gap of a time series analysis platform.

While our approach covers the wide range of tasks related to time series, several discussions exist. FAP proposes a similar approach. We decided to develop our own platform because of the preference of using Python. While Python offers machine learning frameworks such as *scikit-learn*, it proves to be slower than other languages when handling large quantities of data. In our opinion this drawback can be ignored by considering the flexibility of the available machine learning modules. Open source libraries such as *Pandas* and *Numpy* often use other languages in for calculations to speed up the prices. Another point of discussion is the focus of this thesis. This thesis offers a more high level background of machine learning techniques, the platform and time series specific issues. This thesis aims to give a more broad overview instead of a more in-depth knowledge. This choice is motivated by the importance to understand the necessity of the tool and the offered functionality. Not every algorithm is included in the platform. Our modular approach covers this argument and we encourage the user to expand the application with their own modules.

Research is never finished and so is research in time series. As time series research progresses, so should our platform. We can derive several approaches for improving our framework, both development and research.

The modular nature of our application makes expanding the platform simple. Future directions could focus on implementing new algorithms inside ETA. Especially motif discovery algorithm development is encouraged. Several algorithms exist for finding similar sub sequences in time series such as [34] and [47]. Next to expanding the application with modules, the core could be expanded as well. The current application supports the analysis of off-line data sets. As time series is generated in real time, adding stream functionality could provide an interesting opportunity.

Ìn future research, the scalability of the platform can be researched. No research has been done on the performance of ETA when working with large time series. Research should show us the performance of the application when stressed by big data sets.

Bibliography

[1] Time series analysis and forecasting with weka. `http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka`. Accessed: 2016-07-10.

[2] Ziv Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

[3] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD Record*, volume 19, pages 322–331. Acm, 1990.

[4] Eli Billauer. peakdet: Peak detection using matlab. *Eli Billauer's home page*, 2008.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[8] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

[9] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[10] NumPy Developers. Numpy. *NumPy Numpy. Scipy Developers*, 2013.

[11] Pan Du, Warren A Kibbe, and Simon M Lin. Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching. *Bioinformatics*, 22(17):2059–2065, 2006.

[12] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[13] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[14] Craig A Ellis and Simon A Parbery. Is smarter better? a comparison of adaptive, and simple moving average trading strategies. *Research in International Business and Finance*, 19(3):399–411, 2005.

[15] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[17] Vladimir Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1):65–75, 2002.

[18] Chris Fraley and Adrian E Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

[19] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[20] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[21] Thore Graepel. Statistical physics of clustering algorithms. *Technical Report*, 171822, 1998.

[22] Erico Guizzo. How google's self-driving car works. *IEEE Spectrum Online, October*, 18, 2011.

[23] Abeyratna Gunasekarage and David M Power. The profitability of moving average trading rules in south asian stock markets. *Emerging Markets Review*, 2(1):17–33, 2001.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[25] Fredric J Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[26] John D Hunter et al. Matplotlib: A 2d graphics environment. *Computing in science and engineering*, 9(3):90–95, 2007.

[27] Han Jiawei and Micheline Kamber. Data mining: concepts and techniques. *San Francisco, CA, itd: Morgan Kaufmann*, 5, 2001.

[28] jMotif Github. Z-normalization of time series, 2016. [Online; accessed August 20, 2016].

[29] Eric Jones, Travis Oliphant, Pearu Peterson, et al. Scipy: Open source scientific tools for python, 2001. *URL http://www. scipy. org*, 73:86, 2015.

[30] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[31] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. *Data mining in time series databases*, 57:1–22, 2004.

[32] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.

[33] Vladimir Kurbalija, Miloš Radovanović, Zoltan Geler, and Mirjana Ivanović. A framework for time-series analysis. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 42–51. Springer, 2010.

[34] Hoang Thanh Lam, Toon Calders, and Ninh Pham. Online discovery of top-k similar motifs in time series data. In *SDM*, pages 1004–1015. SIAM, 2011.

[35] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[36] M. Lichman. UCI machine learning repository, 2013.

[37] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P Lankford, and Daonna M Nystrom. Viztree: a tool for visually mining and monitoring massive time series databases. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1269–1272. VLDB Endowment, 2004.

[38] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[39] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[40] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[41] DG Mayer and DG Butler. Statistical validation. *Ecological modelling*, 68(1-2):21–32, 1993.

[42] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56, 2010.

[43] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1483–1492. ACM, 2008.

[44] Massoud Metghalchi, Juri Marcucci, and Yung-Ho Chang. Are moving average trading rules profitable? evidence from the european stock markets. *Applied Economics*, 44(12):1539–1559, 2012.

[45] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[46] Abdullah Mueen. Time series motif discovery: dimensions and applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(2):152–159, 2014.

[47] Abdullah Mueen, Eamonn Keogh, and Nima Bigdely-Shamlo. Finding time series motifs in disk-resident data. In *2009 Ninth IEEE International Conference on Data Mining*, pages 367–376. IEEE, 2009.

[48] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[49] John A Nelder and R Jacob Baker. Generalized linear models. *Encyclopedia of statistical sciences*, 1972.

[50] Pranav Patel, Eamonn Keogh, Jessica Lin, and Stefano Lonardi. Mining motifs in massive time series databases. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 370–377. IEEE, 2002.

[51] Karl Pearson. Principal components analysis. *The London, Edinburgh and Dublin Philosophical Magazine and Journal*, 6(2):566, 1901.

[52] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[53] Jos Polfliet. Pandas-profiling. `https://github.com/JosPolfliet/pandas-profiling`, 2016.

[54] David Stephen Geoffrey Pollock, Richard C Green, and Truong Nguyen. *Handbook of time series analysis, signal processing, and dynamics*. Academic Press, 1999.

[55] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[56] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[57] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[58] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[59] Scikit learn. Comparison of clustering methods available in scikit, 2016. [Online; accessed August 20, 2016].

[60] Carlos N Silla Jr and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[61] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[62] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[63] Stephen J Taylor. Modelling financial time series. 2007.

[64] Teach Tough Concepts: Frequency Domain in Measurements. Comparison between time and frequency domain, 2016. [Online; accessed August 20, 2016].

[65] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[66] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[67] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[68] Guido Van Rossum et al. Python programming language. In *USENIX Annual Technical Conference*, volume 41, 2007.

[69] Yurtman Aras. Automated evaluation of physical therapy exercises using multi-template dynamic time warping on wearable sensor signals, 2016. [Online; accessed August 20, 2016].

[70] Jian Zhao, Fanny Chevalier, Emmanuel Pietriga, and Ravin Balakrishnan. Exploratory analysis of time-series with chronolenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2422–2431, 2011.

[71] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

Appendix A

Adding your own module.

For our application we decided to use a modular approach. Therefore it is possible to implement and use your own algorithms inside the framework. This document shows how to successfully create a module. We describe the conventions applied by the framework and finally we show an example of a module.

Depending on the type of module you are creating it is required to apply a standardized class name.

Our framework relies on some standard functions. These functions are mandatory for each module.

- `getName()`
  This function should return a string containing the name of the module.
- `getOptions()`
  This function should return a dict containing the available parameters. These parameters are automatically included inside the GUI. Below we propose a list with possible GUI elements and how to implement them.
  - Dropdown
    To create a dropdown menu one should return a list containing the dropdown items. An example dict entry:
    `'kernel': ['gaussian','tophat','epanechnikov'],`
  - A dropdown containing columns
    To create a dropdown menu containing the columns of the current

Table A.1: Class naming for modules per type.

Type	**Class name**
Preprocessing	`preProcess_Module()`
Feature extraction	`featureExtraction_Module()`
Machine learning	`machineLearning_Module()`
Plot	`Plot_Module()`

    DataFrame it is necessary to return a string with the word columns. A dict entry for a dropdown menu containing *columns*:
    `'column_1': 'columns',`

  - Input field
    To create an input field it is necessary to return a string with the word *float*. A dict entry for an input field:
    `'clusters': 'float',`

- `getOptionDefaults()`
  This function should return a dict containing the default options for the GUI. The key should be the same as the key in `getOptions()`

- `run(self,DataFrames,parameters=None)`
  This function should contain the algorithm. It receives the DataFrames and the input parameters from the framework. A DataFrame should be returned to the platform.

  - `DataFrames` This is a dict containing the DataFrames corresponding to currently loaded data. This dict contains two entries:
    * `DataFrames['maskedData']`
      This DataFrame only contains the selected columns. A possible application for this DataFrame could be to apply a module to multiple selected rows.
    * `DataFrames['data']`
      This contains the complete dataset including the columns *not* selected.
  - `parameters`
    This contains a dict with all the parameters returned inside

These functions shown above are standard for all types of algorithms. For `machineLearning_Module()`, two situations can be derived. First, the situation with unsupervised learning. For a unsupervised learning algorithm (such as clustering) the `run(self,DataFrames,parameters=None)` function could be called. This function should return a DataFrame containing all the measurements and their corresponding cluster.

Supervised learning inside ETA offers more functions. For example, the generated model can be saved for later use. Several other functions need to be implemented to facilitate this functionality.

  - `fit(dataFrames,parameters=None)`
    This function is used to create the model. In a typical machine learning environment the train set is used to train the model. The complete model should be returned to the platform.

  - `predict(dataFrames,model)`
    This function is used to predict values by using the model. Before applying the model, a model needs to be loaded. This can be done by loading a previously generated model or by generating one. The model loaded in the system is passed through the model parameter. The function should return a dataFrame containing the predicted values.
  - `metrics(yPredicted,yTest)`
    Metrics are used to quantify the quality of the model. ETA offers the functionality to display the metrics inside the program. To calculate this metrics the yPredicted and the yTest parameters provide the predicted and the real values. Again, the results should be returned as a dict.

  It is important that this function returns a DataFrame containing the new data. The framework automatically checks if the number of rows corresponds to the source DataFrame. If this is not the case it asks the user if a new DataFrame needs to be created.

In table A.2 we can see an overview of modules and their file locations.The framework automatically searches for `*.py` files at these locations.

Table A.2: Module locations depending on their type.

Type	**Module location**
Preprocessing	`/modules/preprocessing/*.py`
Feature extraction	`/modules/featureExtraction/*.py`
Machine learning classification	`/modules/machineLearning/classification/*.py`
Machine learning clustering	`/modules/machineLearning/clustering/*.py`
Machine learning regression	`/modules/machineLearning/regression/*.py`
Machine learning miscellaneous	`/modules/machineLearning/miscellaneous*.py`
Plot	`/modules/plot/*.py`

```
import pandas as pd
from sklearn.neighbors.kde import KernelDensity

class featureExtraction_Module():
        def __init__(self):
                pass
        def getName(self):
                return "Density Estimator"
        def getOptions(self):
                settings = {
                'algorithm': ['kd_tree','ball_tree','auto'],
                'kernel':['gaussian',
 ↪  'tophat','epanechnikov','exponential','linear', 'cosine']
```

```
                }
                return settings

        def run(self,dataFrames,parameters=None):
                maskedData = dataFrames['maskedData']
                newData = pd.DataFrame()

                for column in maskedData:
                        kde =
↪   KernelDensity(algorithm=parameters['algorithm'],
↪   kernel=parameters['kernel']).fit(maskedData[column].as_matrix().reshape(-1,
↪   1))

                        newData[column] =
↪   kde.score_samples(maskedData[column].as_matrix().reshape(-1,
↪   1))
                return newData
```

Appendix B

Technical documentation.

Below we find the technical documentation of the software. Unfortunately not every function has been successfully documented. This will be done in the near future. For now, the most important class: `guiHandler` has been included. Of course, the code itself is well commented.

ETA Documentation

Release 1.0

L.K. Hopman

Aug 26, 2016

CONTENTS

Contents:

class `core.guiHandler.`**`Handler`**(*gui*)

Attaches signals from GTK to the program.

`addNewDataFrame`(**args*)

`openFileDialog`(**args*)

`openRecent`(**args*)

`printData`(**args*)

`printMaskedData`(**args*)

`removeColumn`(**args*)

`removeDataFrame`(**args*)

`runMissingRows`(**args*)

`runMissingVis`(**args*)

`runPandasProfiling`(**args*)

`saveFileDialog`(**args*)

`selectAll`(**args*)

`selectInvert`(**args*)

`selectNone`(**args*)

`splitTrainTest`(**args*)

class `core.guiHandler.`**`guiHandler`**

Class handling the GUI for the the application. Inherits from filesHandler and plotHandler

`addNewDataFrame`()

This function adds a new df named usergenerated_%time%. It adds the dataframe to the dropdown. :returns: None

`askNewDataFrame`()

This function creates a dialog asking if the user wants to create a new DataFrame.

Returns None

`clearListColumns`()

This function clears the list containing the columns

Returns None.

`columnsMaskerActive`(**args*)

This function handles the selection of the data.

Parameters

- **`buttonActive`** (*`button.`*) – The button concerning the column.
- **`column`** (*`str.`*) – The column to remove or add to selected data.

Returns None.

`createOptionLabel`(*option*, *grid*)

`fillListColumns`(*dataObject*)

Fill the columns list with column

Parameters **`dataObject`** (*`DataFrame.`*) – The data to fill the columnslist with

Returns None

`fillMenuFeatureExtractionModules()`
This function fills the feature extraction menu with the values loaded inside self.extract

`fillMenuMachineLearningModules()`
This function fills the machine learning menu with the values loaded inside self.learning

Returns None – the return code.

`fillMenuPlotModules()`
This function fills the plot menu with modules available inside self.plot

Returns None.

`fillMenuPreprocessingModules()`
This function fills the preprocessing menu with modules available inside self.pre

Returns None.

`fillMenuProfilingModules()`
This function fills the profiling menu with modules available inside self.profile

Returns None.

`getFeatureExtractionModules()`
This function creates the self.extract global and gets all the feature extraction modules. :returns: list – a list containing feature extraction modules.

`getMachineLearningModules()`
This function creates the self.learning global and gets all the machine learning modules. :returns: list – a list containing machine learning modules.

`getMetrics(*args)`
This functions retrieves the metrics from a Machine learning module and shows it in the GUI.

Parameters **module** (*str.*) – The module to use.

Returns None

`getPlotModules()`
This function creates the self.plot global and gets all the plot modules. :returns: list – a list containing plot modules.

`getPreProcessingModules()`
This function creates the self.pre global and gets all the preprocessing modules. :returns: list – a list containing preprocessing modules.

`getProfilingModules()`
This function creates the self.profile global and gets all the profiling modules. :returns: list – a list containing profiling modules.

`getTargetColumn()`

`initMainScreen()`
This function initializes the main window and fills all the submenus with the availablemodules :returns: None

`loadFile(dialogData=None)`
This function opens openFileChooserWindow and handles the returned list. It loads the data in a new df. :returns: None.

`loadModel(*args)`

onDataFrameDropdownChange(**args*)

This function is called on dropdown menu change. It fills the list of columns with the selected DataFrame.

Parameters **combobox** (*combobox.*) – The name to use.

Returns None.

openFileChooserWindow()

This opens the file chooser window for opening a file. :returns: list – containing filename(str), seperator(str) and header (bool).

openModuleOptions(**args*)

This function opens the module options if options exist for this module.

Parameters

- **optionType** – The type of module ('extract','pre','plot','learning')
- **module** – The module to load

Returns None

openRecent()

plotMissingRows()

This function runs the missing rows script which visualizes the missing rows.

Returns None

Raises ValueError, BaseException

predict(**args*)

printData()

This prints the selected DataFrame in the terminal

Returns None

printMaskedData()

This prints the selected DataFrame and selected columns in the terminal

Returns None

removeColumn()

This function removes a column from the dataFrame and updates the GUI column list.

Returns None

removeDataFrame()

This function removes the current selected df. :returns: None

runMissingVis()

This function runs the MissingVis algorithm

Returns None

Raises BaseException, KeyError

runOptions(**args*)

This function gets all setted parameters and runs the module.

Parameters **inputs** (*list.*) – The inputs to use as parameter

Returns None

runPandasProfiling()

This runs pandas-profiler on the selected data

Returns None

Raises AttributeError, KeyError

`saveFile`()
This function opens a saveFileDialog and eventually saves the selected data. :returns: None.

`saveModel`(**args*)

`selectAll`()
This selects all columns in the selected dataFrame

Returns None

`selectInvert`()
This function inverts the selection of all the columns

Returns None

`selectNone`()
This function deselects all columns in the selected dataFrame

Returns None

`showMessage`(*title*, *message*, *messageType='info'*)
This creates a GUI dialog containing an info message

Parameters

- **`title`** (`str.`) – The title of the window
- **`message`** (`str.`) – The error message

Returns None

`showMetrics`(*title*, *metrics*)

`splitTrainTest`()

`updateDataFrameDropdown`()
This function retrieves the current DataFrames and fill the dropdown menus with these DataFrames. :returns: None

class `core.dataFramesHandler.`**`dataFramesHandler`**

`addColumn`(*dfName*, *column*, *data*)

`addDataFrame`(*name*, *df*)

`addToMaskedData`(*dfName*, *column*)

`copyDataToMasked`(*dfName*)

`createEmptyDataframe`(*name*)

`getData`(*dfName*)

`getDataFrames`(*dfName*)

`getListOfFrames`()

`getMaskedData`(*dfName*)

`removeColumn`(*dfName*, *column*)

`removeDataFrame`(*dfName*)

`removeFromMaskedData`(*dfName*, *column*)

resetIndices(*name*)

splitTrainTest(*test_size*, *randomized*, *currentDataFrameName*)

class `core.featureExtractionHandler.`**featureExtractionHandler**(*dfHandler*, *guiObject*)

extract(*parameters=None*)

getOptionDefaults()

getOptions()

listModules()

loadModule(*module*)

runModule(*parameters*, *ready*)

class `core.filesHandler.`**filesHandler**
Class for handling the IO operations

getData()
This function returns the data :returns: list – Containing maskedData and Data

getMaskedData()
This function returns the masked data :returns: list – Containing maskedData

getTimeData()

loadFile(*dialogData*)
This loads the file into a dataframe. :returns: DataFrame – returns a dataframe containing the file

loadModel(*filename*)
This function loades the selected model.

Parameters **filename** (*`str.`*) – The path to the file to load

Returns Model

saveMaskedData(*df*, *saveData*)
This function saves the selected data to the disk

Parameters

- **df** (*`DataFrame.`*) – The dataframe to save
- **saveData** (*`list.`*) – Information such as filepath

Returns None

saveModel(*model*, *filename*)
This function saves the model to the disk

Parameters

- **model** (*`machine learning model.`*) – The model to save
- **filename** (*`str.`*) – Filepath to save the file to

Returns None

class `core.machineLearningHandler.`**machineLearningHandler**(*dfHandler*, *guiObject*)
Class for handling the the machine learning modules

cluster(*parameters*, *ready*)

fit(*parameters*, *ready*)

getMetrics()

getModel()

getName()

getOptionDefaults()

getOptions()

getResultOptions()

learn(*parameters=None*)

listClassificationModules()

listClusteringModules()

listRegressionModules()

loadModel(*model*)

loadModule(*module*)

predict()

runClustering(*parameters=None*)

class core.plotHandler.**plotHandler**(*dfHandler*, *guiObject*)

createPlotWindow()

createToolbarWindow()

getOptions()

getParameters()

getReturned(*returned*)

listModules()

loadModule(*module*)

plot(**parameters*)

plotCorrelationMatrix(*data*)

removePage(*widget*, *child*)

setOptions(*parameters*)

update(*widget*)

class core.preProcessingHandler.**preProcessingHandler**(*dfHandler*, *guiObject*)

getOptionDefaults()

getOptions()

listModules()

loadModule(*module*)

preProcess(*parameters=None*)

runModule(*parameters*)

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX