MASTER THESIS IN MATHEMATICAL SCIENCES

Time Series Active Learning using Automated Feature Extraction

WILLIAM LINDSKOG
Spring 2021

Centre for Mathematical Sciences
Division of Mathematical Statistics
LUND UNIVERSITY, FACULTY OF ENGINEERING
Lund, Sweden

Time Series Active Learning using Automated Feature Extraction
WILLIAM LINDSKOG
Centre for Mathematical Sciences
Division of Mathematical Statistics
LUND UNIVERSITY, FACULTY OF ENGINEERING

Abstract

Time series classification is a prevalent problem for sensor data in an industrial setting. The ability to classify time series correctly allows for predictive maintenance which is the process of optimally maintaining assets and ensuring minimal "down-time" of machines or processes. A requirement for classifying the time series are labeled time series that can be used as training instances. Labelling data is a time consuming and costly activity. Overcoming this, time series active learning attempts to label the most *useful* time series in order for a machine learning model to learn quicker. Time series active learning is argued to be an understudied topic. Time series classification also faces the problem of large data dimensionality. By extracting features from the time series, one may reduce the dimensionality and leverage quicker processing. It is found that both suggested methods are faster and more accurate than a state-of-the-art method for time series active learning.

Keywords: Time Series, Active Learning, Machine Learning, Feature Extraction, Predictive Maintenance

ACKNOWLEDGEMENTS

I would firstly like to thank my supervisor from Viking Analytics, **Oskar Liew**. Thank you for providing me with helpful insights and answering my questions, and especially supporting me through the programming. I would have never been able to complete the code without your insights.

I also want to thank **Davide Badalotti** for great talks and coding experience. You really made implementing scientific code interesting and fun. I must say that I learned a lot from you. You are a great programmer and I hope that I will have the possibility to work with you again. Thank you for your part of the ACTS implementation.

Moreover, I would like to thank my LTH supervisor **Erik Lindström** for the help with my thesis writing. Your insights really helped me structure my work and eventually come to conclusion.

Lastly, I would have not been able to write such an interesting thesis without **Viking Analytics**. You gave me the opportunity to work with interesting real-world solutions and I have learned a lot from working with you.

Acronyms

K-**NN** *K*-Nearest Neighbours. 12, 13

R-**NN** Reverse Nearest Neighbours. 12

ACTS Active Learning method for Time Series. v, 5, 6, 10, 11, 13, 18–22, 26, 27, 29

AI Artificial Intelligence. 1, 2

AL Active Learning. 3–6, 8, 9, 17–20, 23, 24, 26, 28, 29, 35, 36

ALM Active Learning Manager. 18

CPSs Cyber-Physical Systems. 1, 3

DM Data Manager. 18

DS Data Science. 4

IoT Internet of Things. 1–3

LSTM Long-Short Term Memory. 5

ML Machine Learning. 2–4

NN Neural Net. 5

PdM Predictive Maintenance. 2–4

RNN Recurrent Neural Network. 5, 13

SVM Support Vector Machine. 10, 18

TSC Time Series Classification. 3–5, 7, 8, 17

TSFEL Time Series Feature Extraction Library. v, 4, 17–24, 26–29

TSFRESH Time Series Feature Extraction based on Scalable Hypothesis tests. v, 4, 17–29, 35–38

Contents

1 Introduction **1**
- 1.1 Artificial Intelligence in Industry 4.0 . . . 1
- 1.2 Industrial Data . . . 3
- 1.3 Active Learning . . . 3
- 1.4 Feature Extraction . . . 3
- 1.5 Motivation . . . 3
- 1.6 Aim of Thesis and Problem Definitions . . . 4
- 1.7 Delimitations . . . 4
- 1.8 Thesis Structure . . . 4

2 Related Work **5**

3 Theory **6**
- 3.1 Data and Dimensionality Reduction . . . 6
  - 3.1.1 Time Series . . . 6
  - 3.1.2 Feature extraction . . . 7
- 3.2 Active Learning . . . 8
  - 3.2.1 Uncertainty Sampling . . . 8
  - 3.2.2 Diversity Sampling . . . 9
- 3.3 Support Vector Machine . . . 10
- 3.4 ACTS . . . 10
  - 3.4.1 Pattern Discovery . . . 10
  - 3.4.2 Modeling . . . 11
  - 3.4.3 Question Selection . . . 11

4 Data **14**
- 4.1 Production Dataset . . . 14
- 4.2 Ball Bearing Dataset . . . 14
- 4.3 Swedish Leaf Dataset . . . 16

5 Method **17**
- 5.1 Time Series Feature Extraction . . . 17
- 5.2 Active Learning Loop . . . 18
  - 5.2.1 Initializing Active Learning . . . 18
  - 5.2.2 The Loop . . . 18
- 5.3 Comparison . . . 18

6 Results **20**
- 6.1 Duration . . . 20
- 6.2 Accuracy ACTS vs. TSFEL and TSFRESH . . . 20
- 6.3 Confidence Intervals . . . 22
- 6.4 Different Active Learning Methods . . . 24

7 Discussion **26**
- 7.1 Comparison with ACTS . . . 26
- 7.2 TSFEL vs. TSFRESH . . . 26
  - 7.2.1 Benchmark Dataset Comparison . . . 26
  - 7.2.2 Uniform Dataset Comparison . . . 27
  - 7.2.3 Swedish Leaf Comparison . . . 27
- 7.3 Random- and Diversity Sampling . . . 27
- 7.4 Questions . . . 27

8 Conclusion and Future Research **29**

References **30**

9 Appendix **35**

1 Introduction

A decade has almost passed since the term Industry 4.0 was coined and it has been prominent ever since. It is known that humans have experienced major technological advancements through three well studied industrial revolutions (Montero & Blanc 2019) and are now on the brink, if not even experiencing the fourth. This is what is now known as Industry 4.0. It should be mentioned that Industry 4.0 is not only an industrial event but include other areas e.g. biology in form of gene modification (Li et al. 2017). In their article, (Liu & Xu 2017) argue that one may present Industry 4.0 with distinguishable characteristics from its predecessors. Industry 4.0 allows for broad Cyber-Physical Systems (CPSs) in a manufacturing setting. In depth, Industry 4.0 originates in machines' and processes' ability to intelligently act and communicate, utilizing data and information communication. Cloud computing, real-time processes, self-organizing are only a few out of many components that support Industry 4.0 (Vogel-Heuser & Jumar 2019)(Vaidya et al. 2018). Technologies such as Internet of Things (IoT)(Sarma & Girão 2009)(Atzori et al. 2010) and Artificial Intelligence (AI) (Russell & Norvig 2009) allows for real-time integration of monitoring and tracking systems, and intelligent decision making by machines, respectively. Instead of having manual labor attempting to identify flaws, broken parts of machinery, or endlessly monitoring information, an option would be to let computers do the job for humans. Computers may identify essential underlying patterns and structures faster than humans, even if the problem varies e.g. energy efficiency monitoring, flawed regulators (see dataset example in Section 4.1), poor machinery, or even distinguishing leaves (see Section 4.3). The technology itself is not new, nevertheless, both IoT(Dachyar et al. 2019) and AI(Perrault et al. 2019) has seen a relatively recent surge in usage, in terms of research within the field, mainly due to improvements in hardware, calculation speed, network technology, and storage capacity. It is evident that in an industrial setting, these technologies could have huge potential upsides.

Introducing the term *Smart Factory*, which can be seen as a fundamental part of Industry 4.0. In a smart factory one would ensure vertical integration, meaning that a factory's operations is closely linked and intelligently executed (Dorst et al. 2016). Quoting Radziwon et al. (2014), a smart factory is "*... a manufacturing solution that provides such flexible and adaptive production processes that will solve problems arising on a production facility with dynamic and rapidly changing boundary conditions in a world of increasing complexity.*

1.1 Artificial Intelligence in Industry 4.0

As previously mentioned, AI may be seen as one of the key components in achieving smart factories. It is the study in which one seeks to computationally explain intelligent agents. A motivation to pursue development of AI systems is *Autonomy*. An AI application may be seen as autonomous in the sense that it behaves based on training and experience. With training, one refers to the process of learning (Russell & Norvig 2009). AI is an overarching scientific domain, including subfields e.g. robotics, machine learning, computer vision, and natural language processing.

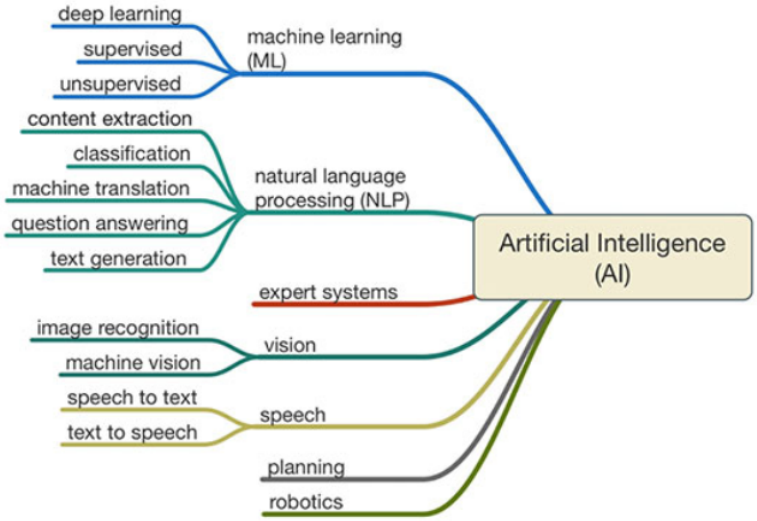

Figure 1: AI as a field with its relevant subfields (Russell & Norvig 2020)

One of the most prevalent subfields of AI is Machine Learning (ML). ML is the study of computer algorithms that improve automatically through experience and by the use of data. Machines' process of learning includes taking empirical data as input and construct helpful models that can help us humans understand underlying patterns or concepts, e.g. regression- or classification models. The stated types of models, also mentioning clustering and anomaly detection, all have in common that they demonstrate what problem they seek to overcome. A classification output could be e.g. demographic segmentation of customers for a large retail chain. Furthermore, ML can be divided based on its learning approach. The learning approaches are:

1. **Supervised learning** - Utilizes annotated/labeled data. The use of experienced professionals for the labelling process is necessary. It is applicable for regression and classification tasks. Consider a customer segmentation example where one already knows if a customer is a teenager or not. Evidently, each customer has therefore received a label; (1) teenager or (2) not a teenager. For this binary classification challenge where labels are given, one trains a ML model in a so called supervised setting. The goal is to let the ML model learn how to distinguish between the two classes based on their labels.

2. **Unsupervised learning** - ML models are required to find patterns, taking raw unlabeled data into account. Clustering is a common method for unsupervised learning. Consider the same customer segmentation example. One wishes to classify whether a customer is a teenager or not, nevertheless only possessing information that is not explicitly the age of the customer. By looking at e.g. spending habits, credit score, online activity levels, one may get a hint of the customers age. In an unsupervised setting, a machine learning model is trained to find these underlying characteristics and may thereafter segment customers based on other parameters.

3. **Semi-supervised learning** - Utilizing a small set of labeled data and/or when few features are known. One advanced and promising kind of semi-supervised learning is reinforcement learning where the learning approach is based on reward and punishment systems. A machine learning model is trained through exploration and exploitation. If it performs well (segments customers with high accuracy) it gets rewards, otherwise it gets "punished". The aim of a reinforcement algorithm is commonly set to maximize a score, thereby attempting to receive as many rewards as possible. If it gets punished, the score decreases (Mitchell 1997).

ML is arguably one key component in a smart factory as it provides solutions to automatically solving and providing interpretations of problems arising in a manufacturing setting (Wuest et al. 2016). It is closely tied with Big Data (De Mauro et al. 2016) as a vast amount of relevant industrial data is necessary in order to provide enough input to a ML system so that it can derive insights from it. This is where IoT plays a vital role for industrial manufacturers, to restructure their organisation into a more data driven and autonomous body (Mourtzis et al. 2016). Industrial ML systems are nowadays integrated into e.g. optimization-, control-, and security processes, mainly lowering costs without affecting quality of production. However, ML in industry faces challenges mainly correlated with what data it takes as input. Data is not always perfect and ready for processing. Instead irrelevant, noisy, and redundant data is often extracted. Also, it is found that knowledge about ML systems is lacking in industry and that education and/or hiring relevant human capital is necessary (Bajic et al. 2018).

Predictive Maintenance

Sharp et al. (2018) claims that ML can support and help develop processes and activities included in a smart factory. Most ML applications found in industry settings are nowadays supervised, followed by unsupervised. It seems that semi-supervised approaches are commencing to increase in form of reinforcement learning applications (Sarker 2021). Examples of ML applications in an industrial setting include downtime minimization, condition monitoring, and failure analysis (Bertolini et al. 2021). These applications all relate to maintenance. Maintenance costs are stated to represent approximately 15 - 70% of total costs in production settings (Bevilacqua & Braglia 2000). Supported by the increase of IoT and AI usage, is Predictive Maintenance (PdM). It leverages on accurate sensor data and forecasting techniques to predict future events and outcomes. It is an integral part of Industry 4.0 and smart factories (Li et al. 2016). PdM explains how components of machines and processes can be analyzed and evaluated on their remaining life time. Consequently, unnecessary maintenance activities can be identified and machine down time may be minimized, therefore reducing costs (Selcuk 2017). Lastly, it is found in literature that a problem arising for PdM is the power necessary for processing big data. There is a need to efficiently make use of important data and filter out unnecessary data (Yamato et al. 2017).

1.2 Industrial Data

What is sometimes known as Industrial Big Data, is supported by IoT as it allows for huge amount of data to be generated and communicated. It helps industries become CPSs and data generated is set to increase over coming years (Mourtzis et al. 2016)(O'Donovan et al. 2015). As previously mentioned, relevant data supports the integration of smart factory applications such as PdM. Nevertheless, it is also accompanied by challenges e.g. reliability and safety. Moreover, as sensors mostly collect data with a frequency of a predetermined time interval, they consequently retrieve data that is subject to a temporal domain. This means that time is an essential parameters for the data retrieved (Yan et al. 2017). Lastly, the sensor data generated and transmitted in a smart factory has been investigated in recent years, while focus has mainly been in acquiring prominent data processing techniques (Xu & Hua 2017)

Time Series

In their article, Christ et al. (2018) argue that IoT, among with other applications, serve to provide advanced ML systems with data that is subject to a temporal domain, known as time series. Time series data may, as previously stated, be generated from sensors and with improvement in (sensor performance and) IoT applications, one can expect the amount of time series data generated to increase. Motivated partly by PdM, recent research has focused on time series forecasting and classification in order to determine current state of machinery of future events (Alexandrov et al. 2020). Time series forecasting is intuitively the action of prefiguring the expected state or outcome of an agent or measurement based onhistorical data. Time Series Classification (TSC) deals with categorizing time series within determined classes e.g. over or under a threshold value. Recent research include modern approaches using new machine learning techniques e.g. artificial neural networks and deep learning (Tealab 2018)(Sezer et al. 2020)(Ismail Fawaz et al. 2019). Peng et al. (2017) imply that time series classification (TSC) indeed shows great potential but that real-world cases are often overlooked. They argue that a common challenge for TSC is the lack of training set as it may be time consuming and expensive to acquire labeled time series data, especially considering the complex nature of time series.

1.3 Active Learning

Peng et al. (2017) states that one challenge for TSC is observed in the expensiveness of labeling time series data. They attempt to utilize Active Learning (AL), in order to tackle this problem. It is also mentioned that time series active learning is an understudied topic. AL is the subfield of ML in which an algorithm is allowed to "choose" data to learn from. AL seeks to find the most *useful* (also referred to as *informative* or *uncertain*), data points to learn from. A key hypothesis is that an algorithm learns quicker this way (Konyushkova et al. 2017). Considering a ML model and a set of unlabeled data, one may utilize AL in order to identify useful data points and later query a so called "Oracle" for the correct label. In practice the oracle is a person with domain expertise. Querying means asking for a data point's label. In practice, the oracle labelling the queried data points is often a person with domain expertise. There are various methods that can be implemented and serve different purposes (Settles 2011).

1.4 Feature Extraction

Motivated by the complex nature of time series and their high dimensionality are studies concerned with efficient extraction of relevant information (Lin et al. 2007). By extracting statistical values, hereby referred to as features, from time series data, e.g. mean value, skewness, autocorrelation, one can accomplish dimensionality reduction that may represent that original data well. Importantly, by acquiring a set of features, one can retain a significant amount of original information contained in the time series. Proper time series feature extraction is therefore suitable for TSC (Krawczak & Szkatuła 2014). Utilizing the lower dimensional representation of the time series data, one can easier perform computations e.g. classification, as it does not require as much data processing (Lei & Wu 2020). Research on what features should be compute has been carried out with varying results. Fulcher & Jones (2017) suggest an extracted feature set of approximately 7700 time series features, while Barandas et al. (2020) suggest a set of little over 60 features.

1.5 Motivation

For a specific problem, there might be one feature that may represent the data well, but for another problem it may not be significant. Identifying an easy-to-follow procedure, capturing a relatively large set of features, that is mostly automated to classify time series data is desirable from an industry point-of-view. It should be

possible to use this approach on different problems in different settings. See Section 4. AL for TSC has received little attention (Peng et al. 2017). Nevertheless, it is highly relevant in many real-world applications of ML. In a future Industry 4.0, PdM is seen as essential for operating autonomous systems. Not only would it allow for speedy analysis of event root cause but it would support professionals in their work even if they are not experienced with Data Science (DS) or ML. This work is deemed necessary for advancing towards Industry 4.0 and utilizing the promise of unlabeled data.

1.6 Aim of Thesis and Problem Definitions

The aim of this thesis is to investigate if automated feature extraction combined with AL can provide a superior overall performance than a state-of-the-art method for TSC using AL. Specified problems include:

- What performance measurements are to be assessed considering the business perspective?
- What data sets should be included to provide generality?
- How should one extract features?
- What features should one extract?
- What active learning methods should one utilize?

1.7 Delimitations

Deep learning, clustering, and representation learning will not be considered. Only statistical feature extraction will be used for TSC. A feature selection step will not be added after having extracted features. Extracting statistical features, Time Series Feature Extraction Library (TSFEL) (Barandas et al. 2020) and Time Series Feature Extraction based on Scalable Hypothesis tests (TSFRESH) (Christ et al. 2018) will be used. HCTSA (Fulcher & Jones 2017) will not be used as it is considered too computationally heavy. The 3 datasets chosen are upper limited by 15 classes but includes binary-, as multi-class setting. Furthermore, for active learning techniques, only uncertainty-, random-, and diversified sampling will be used. Classifiers are limited to Support Vector Classifier. The AL techniques are used through the proprietary MultiViz, created by Viking Analytics[1], builder which constitutes of the modAL (Danka & Horvath 2018) package which is built on sklearn.

1.8 Thesis Structure

Following *Introduction* (Chapter 1) is *Related Work* (Chapter 2) where research relevant to the aim of the thesis is presented. Thereafter, applicable theory can be found in *Theory* (Chapter 3), *Data* (Chapter 4), and thesis methodology in *Method* (Chapter 5). This is later followed by *Results* (Chapter 6), *Discussion* (Chapter 7), and *Conclusion and Future Research* (Chapter 8).

[1]Company at which this thesis work has been conducted

2 Related Work

Extensive research has been pursued for TSC. Classic techniques e.g. KMeans (Kobylin & Lyashenko 2020) and CART (Gocheva-Ilieva et al. 2019) for time series classification has been carried out with varying results. Other attempts to classify time series include the usage of Recurrent Neural Network (RNN) (Dennis et al. 2019), Long-Short Term Memory (LSTM) Convolutional Networks (Karim et al. 2019), and many more (Bagnall et al. 2017). Currently, the Neural Net (NN) classifier, combined with various distance functions, is arguably one of the most popular (Lines & Bagnall 2015). Further extensions to this type of classifier has been conducted, e.g. including the use of ensemble methods, and extracting descriminative patterns of the time series, also known as *shapelets*. HIVE-COTE is a prominent classifier, an ensemble of 35 classifiers with an hierarchical structure (Lines et al. 2018). It is registered to have been tested on 85 datasets from the UCR/UEA archive (the largest repository of time series datasets) (Fawaz et al. 2019). It can be seen as a mix of supervised and unsupervised methods but comes with drawbacks:

1. Many of the ensembles require labeled data which in return is time consuming and expensive.
2. Not only the process of labelling data is time consuming. The algorithm's time complexity is $O(n^2 \cdot D^4)$, where n is total amount of time series provided, and D are the time series' respective length.

The problem of labeled time series is prominent for many TSC techniques. Even though they show great promise in various fields e.g. genomics and physics, providing great accuracy, they fail to meet industrial criteria of fast solutions, mainly due to large time complexity of algorithms and/or time labelling time series (Ismail Fawaz et al. 2019), (van Kuppevelt et al. 2020).

Another interesting approach is clustering time series. It addresses the issue of unlabeled data as it is uncommon that labels are automatically generated. Tavakoli et al. (2020) also choose to highlight the temporal dimensionality of time series data and that they consider proper feature selection to be crucial. They suggest a two-step procedure in which one reduces data dimensionality using statistical feature extraction, thereafter apply clustering method to separate the reduced time series data, and add labels to them. The second part of the procedure utilizes an autoencoder and traditional "train-test" supervised learning approach to measure the accuracy. Their research demonstrates a synthesized way of classifying time series by reconstructing an unsupervised problem into a supervised problem. They apply suggested procedure to financial data which is characterized by volatility and return. This simplifies feature extraction since one is aware of what features must be extracted. If one is unaware of what features to extract, this may add complications. Also, their data set include relatively few organizations and it is not specified the "run-time", time it takes to train models.

Further TSC approaches include statistical feature extraction (Lei & Wu 2020), (Ge & Ge 2016). For these sort of methods one lowers the dimension of time series, so that algorithms can train on less computational heavy data. Information, based on earlier stated argument, would not go lost but rather most of it would be retained. Research has been focused on developing excellent feature extraction methods. They vary with the sheer size of the set of features which supposedly would be extracted, their field of application, and programming language (Christ et al. 2018), (Barandas et al. 2020), (Fulcher & Jones 2017), (Naul et al. 2016), (Nun et al. 2015). They serve as a great contributions to methods attempting to classify time series based on their features. This methods can be considered computationally light as finding certain statistical values e.g. maximum value, is a linear process. Nevertheless, solely by using this method, one does not tackle the problem of labeled data.

He et al. (2016) is one of the first to apply AL to TSC. They define an uncertainty score that can be computed for a time series. The uncertainty of a time series is a distance function that explores the possibility of a time series to belong to one class. They also define a utility score for which a time series relevance to an algorithms training process is computed. The time series are assessed based on their scores and thereafter queried. In Peng et al. (2017), they present Active Learning method for Time Series (ACTS). It computes uncertainty and utility values of deterministic patters identified in time series. At the time of writing it is a state-of-the-art method for active learning for time series classification. It is a tweaked version of the work of He et al. (2016). By this recent argument, ACTS is chosen as a baseline/benchmark for further testing. This thesis' suggested method for time series active learning using automated feature extraction will be compared to ACTS after it has been implemented.

3 Theory

Following section is divided into 3 parts. They reflect the suggested method for time series AL using automated feature extraction, and present relevant theory. First, the nature of the data is presented as time series, from which one may extract and select features. Initially, data and dimensionality reduction will be presented in Section 3.1. Having reduced the dimensionality of the time series data, one thereafter applies AL methods. AL and its relevant methods are presented in Section 3.2. Lastly, the benchmark method ACTS will be presented in detail in Section 3.4.

3.1 Data and Dimensionality Reduction

Dimensionality reduction techniques are often applied to data when it stems from a higher dimension than allowed or computationally affordable. One always has to sacrifice information when lowering the dimensionality, the goal is however to sacrifice as little as possible and contain most of the information. Dimensionality reduction techniques based on feature extraction of time series has already shown promise (see Section 1.4). Lower-dimensional data is thereafter imputed into a system, ready for modelling or other use. It is possible that what seems to be high dimensional data can mainly be constituted out of few variables. Dimensionality reduction can therefore prove to be useful in order to remove redundant information (Carreira-Perpinán 1997). Also, found by Bellman (1957), he states that the sample size needed to estimate a functional value to a certain degree of accuracy, grows exponentially with the addition of extra variables. Dimensionality reduction is therefore highly relevant when dealing with large and complex data sets. Mathematically, one may assume a given dataset

$$X = (x_{n,D}) = \begin{bmatrix} [x_{1,1},\ x_{1,2},\ ...,\ x_{1,D}] \\ [x_{2,1},\ x_{2,2},\ ...,\ x_{2,D}] \\ \vdots \\ [x_{n,1},\ x_{n,2},\ ...,\ x_{n,D}] \end{bmatrix} \in \mathbb{R}^{n \times D} \tag{1}$$

where X consists of n vectors containing D-dimensional data. Moreover, one may also refer to the intrinsic dimensionality d, normally $d \ll D$. In other words, intrinsic dimensionality refers to a dimension to which the data points included in X lie near to, even though they are contained in a D-dimensional space. Reducing the presented dimensionality of X is therefore the action creating a new data set

$$Y = (x_{n,d}) = \begin{bmatrix} [x_{1,1},\ x_{1,2},\ ...,\ x_{1,d}] \\ [x_{2,1},\ x_{2,2},\ ...,\ x_{2,d}] \\ \vdots \\ [x_{n,1},\ x_{n,2},\ ...,\ x_{n,d}] \end{bmatrix} \in \mathbb{R}^{n \times d} \tag{2}$$

without loosing a significant amount of information that is contained in the data points in matrix presented in equation (1) (Van Der Maaten et al. 2009).

3.1.1 Time Series

Data which is subject to a temporal domain, also known as time series, can be described as three types. Firstly, the temporal data can carry invariant information such as the producer of a material unit. Secondly, it may include variant information that changes erratically (e.g. state of producing unit). Lastly, variant information could also be regularly updated (Elmasri & Lee 1998). Assume in a smart factory setting that ever changing values $x_{i,j}(t)$ for system or device i, and sensor j, is updated with repeatedly with interval Δ_t. Including a time series length of $n_t^{(j)}$, we have

$$x_{i,j}(t_1) \to x_{i,j}(t_2) \to ... \to x_{i,j}(t_t) \to ... \to x_{i,j}(t_{n_t{}^{(j)}}) \tag{3}$$

with the difference in time being

$$x_{i,j}(t_t) \to x_{i,j}(t_{t+1}),\ t_{t+1} - t_t = \Delta_t \tag{4}$$

The time series in (3) can be presented as

$$X_{i,j} = (x_{i,j}(t_1),\ x_{i,j}(t_2),\ ...,\ x_{i,j}(t_t),\ ...,\ x_{i,j}(t_{n_t}^{(j)})^\top \tag{5}$$

$$= (x_{i,j,1},\ x_{i,j,2},\ ...,\ x_{i,j,t},\ ...,\ x_{i,j,n_t^{(j)}}))^\top \tag{6}$$

Usually, in a smart factory setting, one must consider many devices $i \in \{1, 2, ..., m\}$, and multiple sensors $j \in \{1, 2, ..., n\}$, also including the $n_t^{(j)}$ length of the time series. The time series at hand are of multivariate character, more precisely $n \times m \times \sum_{j=1}^{m} n_t^{(j)}$, and it is the total amount of captured values (Christ et al. 2018).

Time Series Classification
Assume, all time series $X(t)$ have accurate labels Y, forming pairs (X, Y) that can be included in a data set of pairs

$$D = \{(X_1(t), Y_1),\ (X_2(t), Y_2),\ ...,\ (X_n(t), Y_n)\} \tag{7}$$

Given a new input time series X, TSC refers classifying this input into class Y, its so called label (Lei & Wu 2020).

3.1.2 Feature extraction

Instead of either applying deep learning or clustering strategies for TSC problems, it is possible to utilize a feature-based approach. As in equation (1) and (2), the dimensionality of time series can be reduced by retrieving statistical features. This process, often referred to as *feature mapping*, aims to present time series as stateless, only allowing features to be presented. Mathematically, feature mapping of statistical feature k for a one-dimensional time series can be presented as

$$\theta_k \ :\ \mathbb{R}^{n_t^{(j)}} \to \mathbb{R} \tag{8}$$

Nevertheless, consider n_f different time series feature mappings from $m \times n$ time series from m devices and n sensors. The resulting feature matrix is $\Theta \in \mathbb{R}^{m \times n_\phi}$, with m rows and $n_\phi = n \times n_f$ columns. where m is the time series collected by each device and $n_t^{(j)}$ is the features that are mapped. An intuitive example is attempting to map the minimum value a time series takes

$$\theta_{min}(X_{i,j}) = \min_x \{x_{i,j,1},\ x_{i,j,2},\ ...,\ x_{i,j,n_t^{(j)}}\} \tag{9}$$

Other more non-intuitive features that can be mapped are e.g. autocorrelation

$$\theta_{autocorrelation}(X_{i,j}, l) = \frac{1}{n\sigma^2} \sum_{t=1}^{n-l} (X_t - \mu)(X_{t+l} - \mu) \tag{10}$$

where l is the lag of time series X, with variance σ^2 and mean μ. When referring to a feature, the minimum value is one out of many features. Further features are e.g. mean, median, max, or min (see Figure 2).

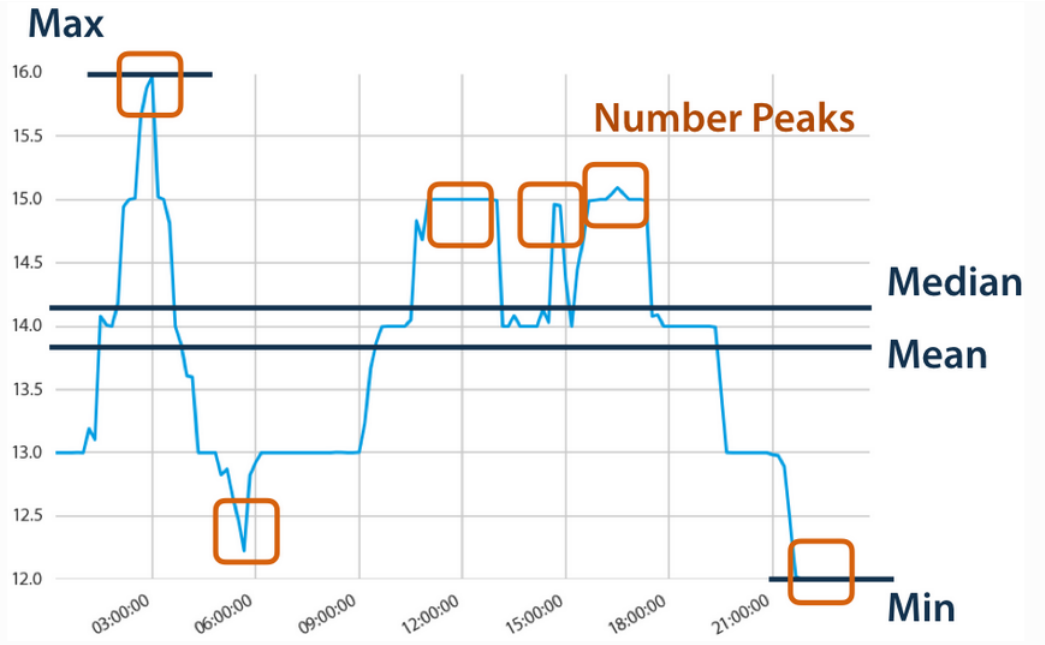

Figure 2: Example of features that can be mapped from an ordinary time series. More features could potentially be mapped, nevertheless, the examples included are quite intuitive (Christ et al. 2018)

Feature extraction from time series is nothing new. Extracting common basic first- and second order features, also known as first and second moments, e.g. maximum, minimum, skewness, and kurtosis for TSC is presented by Nanopoulos et al. (2001). For specialized data e.g. audio data and vibrations, one may calculate peak features (Mierswa & Morik 2005) and/or wavelet-based features (Yen 2000). Nevertheless, a common framework for calculating time series features has long been missing and desired by various interdisciplinary fields. Fulcher & Jones (2014) introduced a framework for initially extracting over 9000 features from time series. Their extensive work has been followed by similar attempts to map significant features that may characterize time series in a lower dimensional space.

3.2 Active Learning

It is nowadays relatively easy to retrieve data from industrial applications. However, increase in processed data outpaces data point labelling operations. Consequently, there may be tons of untapped data that form underlying structures and patterns which are not identified, just due to the huge amount of raw data. (Witten et al. 2011) Modern machine learning strategies e.g. deep learning, requires a significant amount of training data which sometimes might prove to be costly to retrieve (Goodfellow et al. 2016). Historically, feature engineering, deciding what features to use when constructing machine learning models, has been one of the most time consuming activities. Labelling is however proving to be challenging task and attempts to synthesize training data have been made (Bach et al. 2017). Another way of approaching this problem is through the use of AL. Assuming one wishes to use a machine learning model and currently possesses a group of raw, unlabeled data. A pool of data can be quite large and it may certainly contain redundant and unnecessary information. Applying AL techniques, one attempts to find what data point should be labeled next, querying a so called *Oracle* to label the data point which is most *useful*. The oracle is as previously explained in Section 1.3, a person with domain expertise. Querying is selecting the question to ask the oracle. This is repeated until a ML model can achieve satisfying performance. Konyushkova et al. (2017) describe a binary classification problem, with the data set

$$Z = \{(x_1, y_1),\ (x_2, y_2),\ ...,\ (x_N, y_N)\},\ x_i \in \mathbb{R}^N,\ y_i \in \{0,\ 1\} \tag{11}$$

using a standard AL approach, where t denotes a specific iteration ($t \geq 0$):

1. Let the algorithm begin with a small labeled training dataset $L \subset Z$, and leaving a large set of raw unlabeled data $U_t = Z \setminus L_t,\ t = 0$
2. Train classifier C_t using the labeled dataset L_t
3. Allow a specific AL technique to strategically query for a data point (i.e. the most useful data point) $x^\star \in U_t$ to be labeled in the next iteration
4. $x^\star$ is given the label $y^\star$ by the Oracle, and both U_t and L_t are updated
5. t increases and steps $2 \rightarrow 5$ are taken repeatedly

The overarching goal of an AL technique is to find the most useful data points to label in order to speed up the process of creating precise machine learning models (Cohn et al. 1996). There are various techniques that can be applied, and some are discussed in further reading.

3.2.1 Uncertainty Sampling

Considering a collection of data instances, uncertainty sampling intends to compute the *usefulness* of all data instances in the collection and thereafter choose to proceed with the most uncertain instance. There are various methods one can consider when computing the usefullness. Common approaches include:

Classification Uncertainty Classification uncertainty is defined for data point x as

$$\mathcal{U}(x) = 1 - \mathbb{P}(\hat{x}|x) \tag{12}$$

where $\hat{x}$ is the most likely prediction of the specified data point. In other words, the classification uncertainty is the uncertainty of $\hat{x}$ being the prediction given data point x.

Classification Margin The definition of classification margin for data point x is

$$\mathcal{M}(x) = \mathbb{P}(\hat{x}_1|x) - \mathbb{P}(\hat{x}_2|x) \tag{13}$$

more specifically, the difference between the most ($\hat{x}_1$)- and second $\hat{x}_2$ most likely classes. It is in the algorithm's interest to proceed with the smallest margin.

Classification Entropy Lastly, introducing classification entropy as

$$\mathcal{E}(x) = -\sum_k \left(p_k(x) \cdot log(p_k(x))\right) \tag{14}$$

p_k is the direct probability of sample x existing in class k. It can be noted that the calculated entropy is proportional to the average number of guesses taken in order to derive the correct class for x. Moreover, a uniform distribution of samples result in higher entropy and more "guesses" must be made (Lewis & Catlett 1994).

3.2.2 Diversity Sampling

Uncertainty sampling allows one to address where the model is confident or not confident. Diversity sampling looks outside the model in order to find unlabeled data points to label, also using human labelling. See Figure (3)

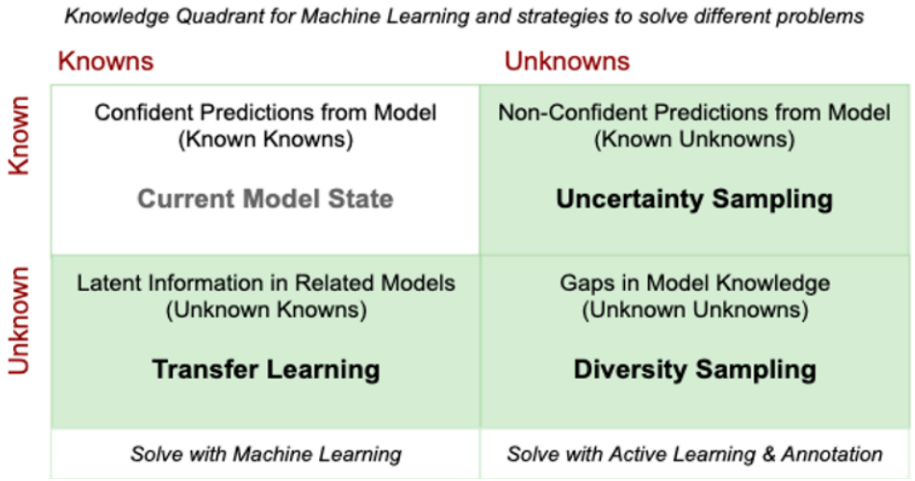

Figure 3: Knowledge quadrant for machine learning. The different areas include variety of strategies one can apply to problems at hand in order to construct the most suitable model (Monarch 2021)

In other words, diversity sampling seeks to explain what one *does not know what one does not know*. Diversity sampling in its nature is mainly exploratory which suits machine learning objectives, constructed for changing environments (Monarch 2021).

Cluster-Based Sampling
One diversity sampling strategy is called *Cluster-based sampling*. In common active learning techniques, one data point is queried meanwhile for cluster-based sampling, a cluster is queried and labeled by an oracle. The cluster itself is represented by n data points that are the most useful, thereby the most uncertain data points. If one were to compare this method with the AL method described in Section 3.2, cluster-based sampling adds clustering of data points in step 3. This method is often combined with uncertainty sampling, therefore resulting in a method where one queries the all instances in a cluster that include the most uncertain instances. It is often carried out to achieve a balance between exploration and exploitation. This method is generally faster than normal uncertainty sampling as n instances are labeled simultaneously and the model does not have to update as frequently (Urner et al. 2013).

Random Sampling
Random sampling, also known as *Simple random sampling* (Bhardwaj 2019)(Turner 2020) can be considered a sort of diversity sampling. Data point x_i in a sample $S = \{x_1,\ x_2,\ ...,\ x_N\}$ is selected randomly, with a probability of

$$\mathbb{P}(x) = \frac{1}{N} \tag{15}$$

Letting random sampling act as a benchmark to various active learning techniques has been used in research. It proves useful as one may conclude how much faster an active learning strategy accomplishes desirable machine learning accuracy. Also, if an active learning technique performs worse than random sampling is not useful (Ilhan & Amasyalı 2014).

3.3 Support Vector Machine

Support Vector Machine (SVM) can be described as an efficient hyperplane searching technique. The objective of a SVM is to identify a hyperplane that efficiently classifies data points, with maximum distance to the data points (margin).

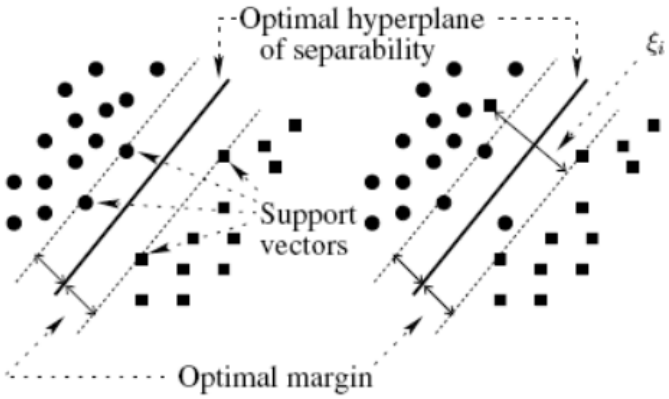

Figure 4: Support Vector Machine that classifies two classes. The optimal hyperplane is shown together with optimal margin (Tzotsos & Argialas 2008)

The binary classification problem can be described as

$$\min_{\omega,b,\zeta} \frac{1}{2}||\omega^2|| + \lambda \sum_{i=1}^{n} \zeta_i \qquad s.t.\ \ y_i(\omega^\top \phi(x_i) + b) \geq 1 - \zeta_i \tag{16}$$

where ζ_i denotes the distance to the correct margin and $\zeta_i \geq 0$. λ is the regularization parameter, $||\omega||$ is the normal vector, $\phi(x_i)$ denotes the transformed input space vector, b is the bias parameter, and y_i is the i^{th} label. The most intuitive example is a hyperplane "cutting through" two classes (binary setting) (Cortes & Vapnik 1995). For classification problems including more than 2 classes, the procedure is to break the problem down into many binary classification problems, either one class versus another, or one class versus the rest (Franc & Hlaváč 2002) .

3.4 ACTS

Presented in Peng et al. (2017) is ACTS which will act as the benchmark method. ACTS is a state-of-the-art method for time series active learning. It builds on time series pattern discovery, uncertainty and utility computation (see Section 2), and a "greedy" selection algorithm. Also, it uses k-nearest neighbours as a classifier. K-nearest neighbour aims to segment data points into k segments based on their relative distance from each other (Cunningham & Delany 2007). Following text summarizes the algorithm in more detail.

3.4.1 Pattern Discovery

In their article, Peng et al. (2017) reason that pattern discovery is a critical problem and attempt to adapt the idea of *shapelet discovery*. Shapelet discovery supports the idea of finding discriminative patterns in a time series (Ye & Keogh 2009).

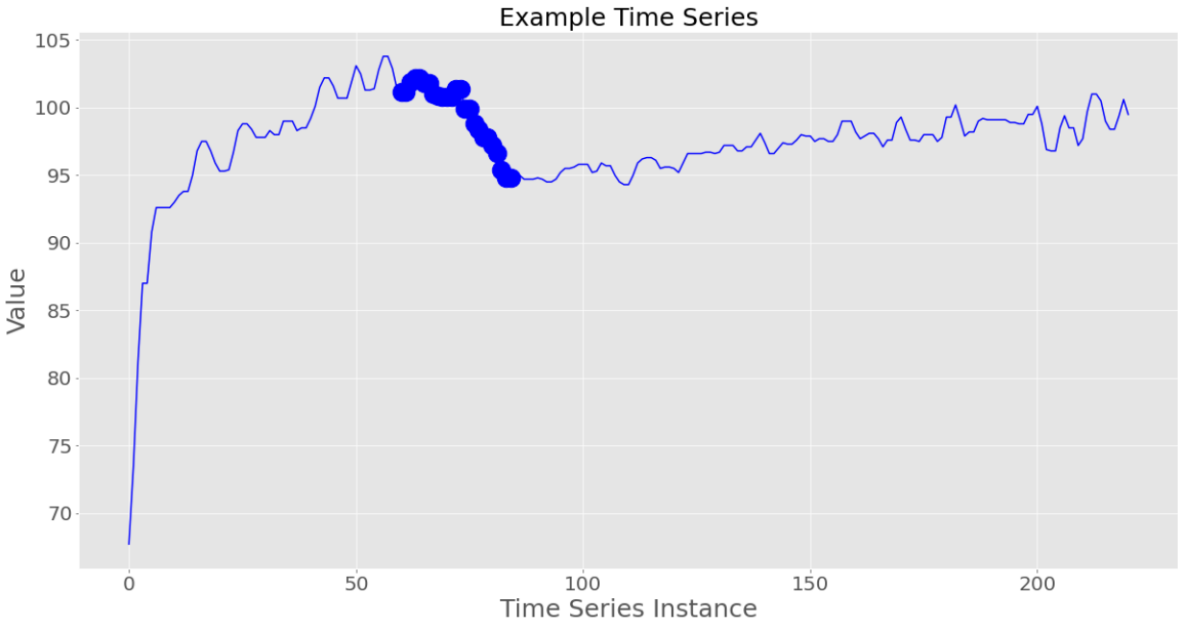

Figure 5: Example of discriminative pattern (filled interval) of a time series. The time series is actually taken from one of the datasets included in this thesis (see Section 4.1)

Shapelets
Firstly, one collects a small training set in which all time series are treated as patterns. New processed time series are assigned to nearest already contained pattern. If patterns were to contain various labels, a patterns splitting method is applied in order to retrieve smaller but still significant patterns. For this, one must define the entropy of a pattern pt as

$$Ent(pt) = \sum_{l \in L} -f(l) \cdot \log(f(l)) \tag{17}$$

where L is a label set, and time series $X \in pt$. The time series in pt having label l are shown as a fraction through $f(l)$. Having performed proper pattern splitting, one seeks to compute the so called *Information Gain*. The information gain could be seen as the improvement measurement of a splitting strategy. What results in the highest information gain is the optimal splitting strategy. Assuming that a pattern split results in pt_1 and pt_2 the information gain is

$$IG(s) = Ent(pt) - \frac{|pt_1|}{|pt|} Ent(pt_1) - \frac{|pt_2|}{|pt|} Ent(pt_2) \tag{18}$$

3.4.2 Modeling

In order to include active learning in ACTS it is necessary to construct probabilistic models on patterns, labels and time series. Using the exponential distribution, the likelihood of time series X given a pattern pt is

$$\mathbb{P}(X|pt) = e^{-\lambda Dis(X,pt)} \tag{19}$$

where the distance function $Dis()$ is the minimum euclidean distance between a time series and a pattern, in this case. Also, for each label l, one can model the conditional likelihood of a pattern using the multinomial distribution

$$\mathbb{P}(pt|l) = Multi(p_1,\ p_2,\ ...,\ p_L) \tag{20}$$

3.4.3 Question Selection

The second part of the algorithm aims at calculating, for each unlabeled instance, its **uncertainty** and **utility** in order to form the value *Question Informativeness*. The time series with the largest value for question informativeness are chosen to be queried. Computing the usefulness[2] of time series data includes calculating

[2]In Peng et al. (2017) they refer to *Informativeness*

the *uncertainty* of a data point, and its *utility*. By calculating the uncertainty, one can assess the classifier's ability to classify a data point correctly. The utility measure exploits the correlation between time series. For highly uncertain data points, there is a risk that they include much redundancy and outliers. Firstly, one must define what is already retrieved

1. **Labeled time series** - The labeled examples until now are referred to as Y_i and they are contained inside the set D_L. Furthermore, each time series is assigned to a pattern, we'll refer to this patterns as pt_{Y_i}
2. **Unlabeled time series** - One may refer to them as X_i, contained inside the set D_U.
3. **Patterns (or shapelets)** - Short subsequences that represents part of the time series. Each time series is assigned to a pattern. Each pattern has its own λ parameter, which is used to calculate the quantity $P(Y|pt)$. Also, each parameter has its own $P(pt|y = l)$ for each l inside the possible labels.
4. **Distance function** - Minimum euclidean distance between time series, or time series and patterns
5. **Label set** - L, i.e. the list of all possible labels inside the dataset. If the classification is e.g. binary the label set will be $L = \{0, 1\}$.

Uncertainty calculation
Firstly, one calculates the uncertainty related to an instance $X \in D_U$. The steps are: **for each instance X**:

- Calculate its K-Nearest Neighbours (K-NN) in D_L. Refer to these as $\{Y_j\}_{j\in 1...k}$, so $Y_1, \ldots, Y_k$ where Y_1 is the closest instance and Y_k the farthest of the K-NN. From this, one must also retrieve the following quantities

$$d_1 = dis(X, Y_1) \tag{21}$$

$$d_k = dis(X, Y_k) \tag{22}$$

- Calculate, for all possible $l \in L$ and using that $Y_j.pt$ being the pattern of a specific time series Y_j, the quantity

$$\bar{P}(y = l|X) = \sum_{Y_j \in \{Y_1,\ldots,Y_k\}} P(X|pt_{Y_j}) \cdot P(pt_{Y_j}|y = l) \tag{23}$$

- Calculate the normalizer

$$Z = \sum_{l\in L} \bar{P}(y = l|X) \tag{24}$$

- Normalize the quantities for the recent calculations using Z and thereafter obtaining

$$\hat{P}(y = l|X) = \frac{1}{Z}\bar{P}(y = l|X) \tag{25}$$

- Finally, calculate the uncertainty for the unlabeled instance X

$$Uncert(X) = \sum_{l\in L} \hat{P}(y = l|X) \log\left(\hat{P}(y = l|X)\right) \frac{d_1}{d_k} \tag{26}$$

Utility calculation
The utility must be calculated for each instance $X_i \in D_u$, but needs the so called Reverse Nearest Neighbours (R-NN) of X, which are the time series $\{Y_1, \ldots, Y_n\}$, that have X_i as a K-NN. To do this

- Calculate, for each time series $Y_i \in D_L$, it's K-NN in D_U.
- For each time series $X \in D_U$, one must check all the time series $Y_j \in D_L$ that have X_i as a K-NN.

This way one obtains the set of R-NN of X_i, which are called $RN(X_i)$. Thereafter, identify the time series in this set with $\{Y_1, \ldots, Y_k\}$. Now, calculate the following quantities

$$dis(X_i, Y_j) \qquad \text{for } Y_j \in RN(X_i) \tag{27}$$

Of which one calculates

$$mDis(X_i) = \max_j dis(X_i, Y_j) \qquad \text{for } Y_j \in RN(X_i) \tag{28}$$

Thereafter,

$$SimD(X_i, Y_j) = 1 - \frac{dis(X_i, Y_j)}{mDis(X_i)} \qquad \text{for } Y_j \in RN(X_i) \tag{29}$$

The first part of the utility is done, one must now calculate the second part which starts by computing the nearest neighbors of X_i in D_L. They are referred to as $\{Y_1, \ldots, Y_k\} \in LN(X_i)$. It is thereafter necessary to compute the following quantity for each possible pattern

$$\Psi(X_i, pt) = \sum_{Y_j \in LN(X_i)} P(X_i|Y_j.pt) I(Y_j.pt = pt) \qquad \text{for } pt \in patterns \tag{30}$$

where the first term is the same of equation (23)'s first part. The second part of the previous equation is on the other hand is defined as

$$I(Y_j.pt = pt) \begin{cases} 0 & \text{if } Y_j.pt \text{ is not the same as } pt \\ 1 & \text{if } Y_j.pt \text{ is the same as } pt \end{cases} \tag{31}$$

The values of $\Psi(X_i, pt)$ are calculated for each pt, and one must also compute the normalizer

$$Z(X_i|PT) = \sum_{pt} \Psi(X_i, pt) \tag{32}$$

What is retrieved is

$$P(X_i|pt) = Z(X_i|PT)^{-1} \Psi(X_i, pt) \tag{33}$$

This element is a core element for the calculation of the utility. It must also be defined for a labeled time series $Y_i \in D_L$:

- Find the K-NN of Y_j in D_L, which are referred to as $LN(Y_j)$.
- Calculate for all patterns pt

$$\Psi(Y_j, pt) = \sum_{Y_k \in LN(Y_j)} P(Y_j|Y_k.pt) I(Y_j.pt = pt) \quad \forall\ pt \in patterns \tag{34}$$

- Calculate the normalizer as

$$Z(Y_j|PT) = \sum_{pt} \Psi(Y_j, pt) \tag{35}$$

- Calculate

$$P(Y_j|pt) = Z(Y_j|PT)^{-1} \Psi(Y_j, pt) \tag{36}$$

One may suppose that all the possible patterns are $[pt_1, pt_2, \ldots, pt_n]$, therefore defining

$$P(X_i|PT) = [P(X_i|pt_1), \ldots, P(X_i|pt_n)] \tag{37}$$

$$P(Y_j|PT) = [P(Y_j|pt_1), \ldots, P(Y_j|pt_n)] \tag{38}$$

Introducing the following quantity which is to be calculated

$$SimP(X_i, Y_j) = 1 - JSD(P(X_i|PT), P(Y_i|PT)) \tag{39}$$

Where JSD is the Jensen Shannon Distance between the arrays. Thereafter,

$$Sim(X_i, Y_j) = SimD(X_i, Y_j) \cdot SimP(X_i, Y_j) \tag{40}$$

and lastly, once everything is defined, still considering that $RN(X_i)$ is the set of RNN in D_L of $X_i \in D_U$, one defines the utility of a time series as

$$Uti(X_i) = \sum_{Y_j \in RN(X_i)} Sim(X_i, Y_j) \tag{41}$$

The final step is to compute the question informativeness

$$QI(X_i) = Uti(X_i) + Uncr(X_i) \tag{42}$$

and select the n most informative instances to query. It is stated in relevant article that the time complexity for the ACTS algorithm is $O(N^2(l + \log(n)))$ where there are N time series with length l.

4 Data

There are three datasets that are considered, the main reason being to analyze implemented algorithms' performance on datasets with varying number of classes. It is worth mentioning that one of the datasets, *Swedish Leaf*, is used in Peng et al. (2017). If received accuracy after having programmed their algorithm scientifically, aligns with their presented accuracy, one may assume that the implementation is more or less correct. The datasets include 2 classes, 4 classes, and 15 classes (Swedish Leaf, the common dataset).

4.1 Production Dataset

The production dataset originates from one of the customers of Viking Analytics. Due to confidentiality reasons, the name of the organisation is hidden but the data itself comes from a production process, also with a frequency of $25Hz$. The classification setting for the data is binary, which means that one seeks to classify the time series data into two classes. Figure 6 shows examples of the different classes.

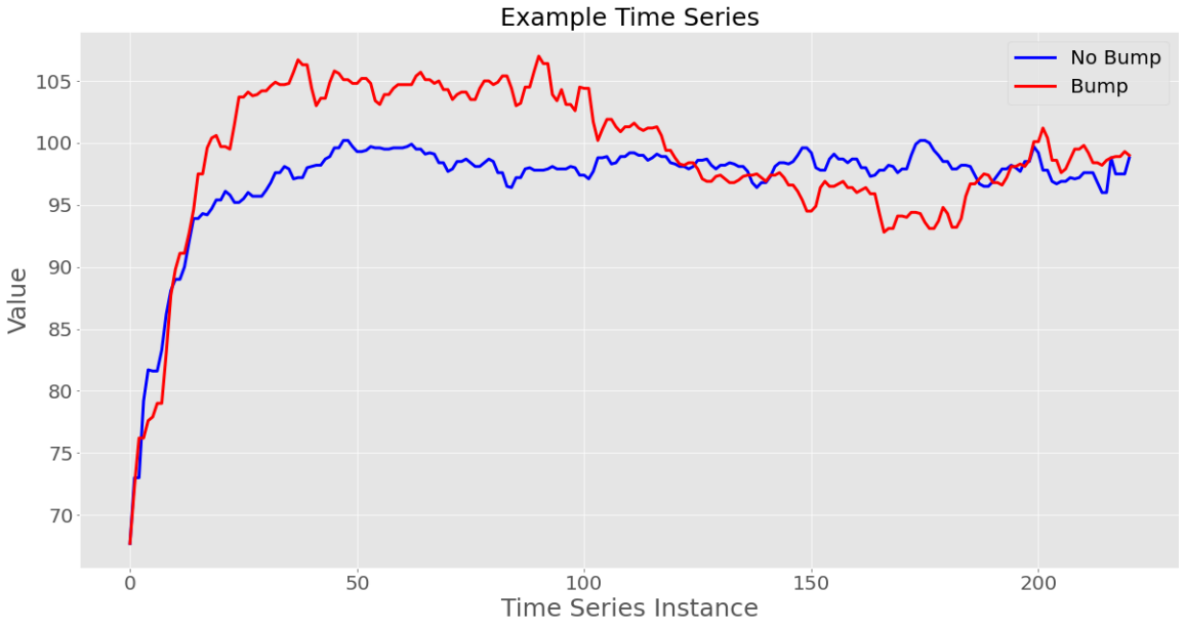

Figure 6: Time series indicating no bump or bump

Padding was carried out in order to work with series of equal length. The main reason for this is that it is easier to work with "same-length" sequences as functions may require this. Common for the time series was identified to be their start value, in this case 66.00. The padding is therefore carried out in the beginning, adding more values of 66.00 to the start of the series. It was reasoned the it should not affect the overall result extensively. Lastly, the time series had to be labeled for further process. 2000 time series were labeled with no bump or bump, the respective labels for the different time series being $[bump,\ no\ bump] = [0,\ 1]$.

4.2 Ball Bearing Dataset

The Ball Bearing dataset[3] includes four classes[4] and a total of 5077 time series. In Smith & Randall (2015), they attempted to create a benchmark for vibration-based rolling element bearing diagnostics. Rolling elements bearings (REBs) are foundational parts of a rotating machine, and their inability to function is one of the most common reasons for machine breakdown. For REBs, it is possible to localize faults by studying impulse responses in an acceleration signal. The impulse response is a direct effect of a fault, and the location may therefore be derived. For the diagnostics, one mainly seeks to identify the so called *envelope signal* as it usually contains clearer fault signals. The envelope signal is identified through amplitude demodulation.

[3]The name stems from the study conducted by Smith & Randall (2015)

[4]The actual data used for this thesis is taken from https://csegroups.case.edu/bearingdatacenter/pages/download-data-file

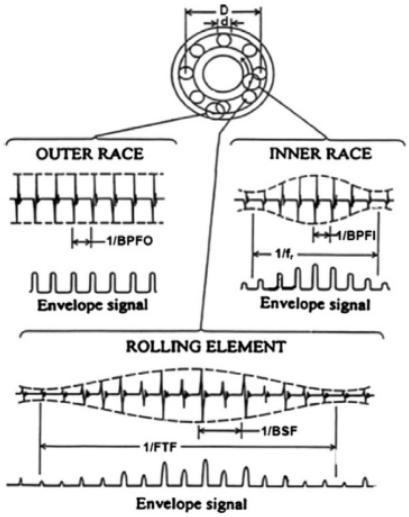

Figure 7: Typical signals and envelope signals from local faults in rolling element bearings (Randall & Antoni 2011)

Their test setup can be found in (Smith & Randall 2015), but shortly described it includes an electric motor with faults being seeded on the drive- and fan-end bearings. However, data used from their test set up has been set to labels [0, 1, 2, 3].

- Label 0 : Baseline data for 1797 rpm
- Label 1 : 12k drive end data, fault diameter of 0.014" - motor load 0 horse power (hp) - motor speed 1797rpm - inner race
- Label 2 : –|||– ball
- Label 3 : –|||– outer race

Each label is one long sequence, which is split into minibatches of 120 data points, forming the dataset. Example of different time series can be found below in Figure 8

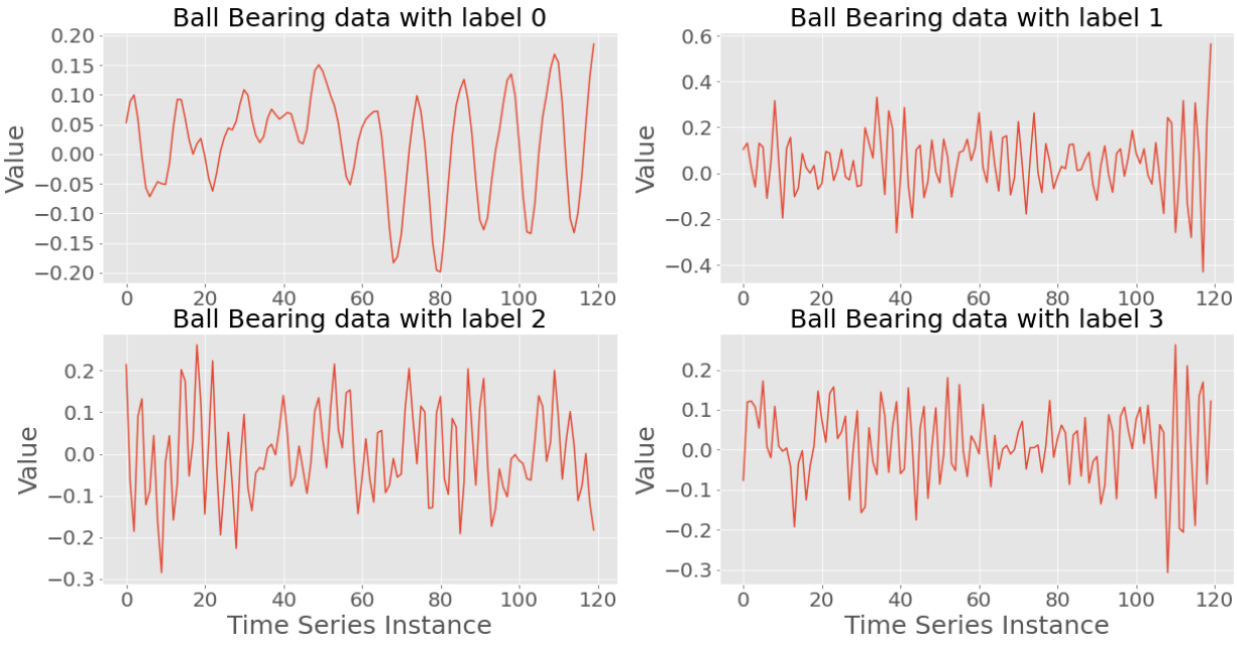

Figure 8: Ball Bearing data set - time series with different labels (example)

4.3 Swedish Leaf Dataset

The *Swedish leaf* dataset, constructed by Söderkvist (2001), contains outlines of various leaves and is a common dataset used for time series classification. Tree classification is mainly constituted by the shape of the leaves. It includes 15 classes, which all consist out of 75 data points, in total 1125 time series. A leaf outline can be illustrated as a time series, see figure below:

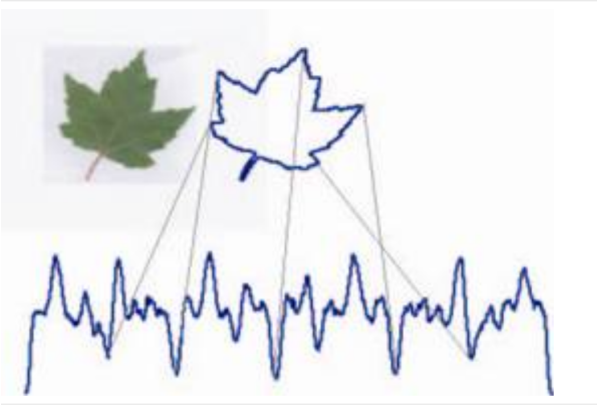

Figure 9: Time series representation of a leaf (Söderkvist 2001)

In Söderkvist (2001), it is reasoned that leaf shape is a great characteristic to investigate for a classification problem. The 15 different sorts of leaves can be found below:

Figure 10: Different types of leaves. From top left corner to bottom right corner: (1) Ulmus carpinifolia, (2) Acer, (3) Salix aurita, (4) Quercus, (5) Alnus incana, (6) Betula pubescens, (7) Salix alba 'Sericea', (8) Populus tremula, (9) Ulmus glabra, (10) Sorbus aucuparia, (11) Salix sinerea, (12) Populus, (13) Tilia, (14) Sorbus intermedia, (15) Fagus silvatica

5 Method

Python is the chosen programming language in which further explained methodology is carried out. It is an open-source programming language, meaning that one may use it and distribute it freely, including for commercial purposes. The licence that is available is governed by the Python Software Foundation (Van Rossum & Drake Jr 1995). Nevertheless, following the methodology presented, one could eventually be able to implement it in other programming languages. The datasets presented in Section 4 includes time series of different characteristics, with varying number of classes, and are used for TSC. It is also worth mentioning that all dataset come with labels. In practice however, one would essentially commence with unlabeled data points.

5.1 Time Series Feature Extraction

Building on Section 3.1, it is desirable to reduce the dimensionality of the time series data using feature mapping. Provided with a time series, one may calculate various statistical features, as in equation (9) and (10), and store them in suitable manner, see Figure 11 for example. Assuming that one possesses many time series, feature extraction can be carried out sequentially for each one of them, resulting in what is known as a feature matrix. The time series are now represented by a feature matrix on which one continues to operate on.

$$\mathbb{X} \in \mathbb{R}^{t \times n_\phi} \tag{43}$$

where t denotes timestamps for a specific sensor and n_ϕ are the features calculated for each time series (timestamp). Having calculated certain features from time series may look like

value__has_duplicate_max	value__has_duplicate_min	value__has_duplicate	value__sum_values	value__abs_energy
0.0	0.0	0.0	0.290392	0.001766
0.0	0.0	0.0	0.022239	0.000506
0.0	0.0	0.0	0.054796	0.001106
0.0	0.0	0.0	0.042157	0.000932
0.0	0.0	0.0	-0.024980	0.000525

Figure 11: Example of calculated features from time series stored in a dataframe, constituting a feature matrix. Each column represent a specific feature which has been extracted from a given time series

Time series features are extracted based on two automatic feature extraction frameworks; Christ et al. (2018) and Barandas et al. (2020). Using TSFRESH, created by Christ et al. (2018) allows for 111 statistical features to be calculated[5], meanwhile the number for TSFEL, created by Barandas et al. (2020), is 60 extracted features[6].

The core methodology is to let a certain number of mathematical calculations be carried out over a set of time series. This can be implemented in any programming language, see (Fulcher & Jones 2017) for reference. In order to calculate the features using python packages presented, it is necessary to preprocess the data to fit the input format. This requires giving each time series an ID or similar and transform it in dataframe format. Before extracting the features, one must save the labels and time series separately, as labels are used later in this process and should not contribute to feature extraction. They should not be imputed with the time series as it would ruin the idea of time series AL. Having received a feature matrix after feature extraction, this matrix is stored with respective labels. The labels can be seen as an extra column at the end of the dataframe. Importantly, each row in the e.g. Figure 11 still correspond to respective label. The order is not affected by feature extraction.

Before commencing the AL part of the method one must look for missing values in the feature matrix. Not a Number, also known as *NaN*, and infinite values are replaced with average and extreme values respectively. Also, if one calculated feature, represented by a column, only consists of infinite values, these are all set to zero. Having done this, one may proceed with the next step. The methods utilizing TSFEL and TSFRESH are hereby only referred to as TSFEL and TSFRESH respectively.

[5] https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html
[6] https://tsfel.readthedocs.io/en/latest/descriptions/feature_list.html

5.2 Active Learning Loop

5.2.1 Initializing Active Learning

After having imputed values, the AL loop begins. As a described AL procedure in 3.2, one initially provides the extracted features with a small set of respective labels. For the production- and ball bearing datasets, 10 labels are given initially while for Swedish leaf dataset, 25 are given. Also, it is ensured that at least one of each labels are given. Thereafter, the extracted feature dataframe and the array containing the labels are shuffled, not loosing connection with respective part. One may here specify how big train and test sets should be.

5.2.2 The Loop

Further process is derived from (Danka & Horvath 2018). The loop begins by choosing a location where to store the feature matrix and the initially given labels. This object could be referred to as a Data Manager (DM). This object should be able to present what parts of the feature matrix that are yet to be labeled. Moreover, one could create a following object called Active Learning Manager (ALM), which takes the data stored in DM and may query the oracle to label the most useful rows of extracted features. This consequently means that the ALM may take various AL query methods as input parameters in order to query for the most useful instance(s). Moreover, SVM has been chosen to act as classifier. The ALM should be able to fit to data given and be able to update itself after having had the oracle label a queried instance. With updating means updating the pool in which unlabeled instances are present. Lastly, the ALM should be able to predict the labels of e.g. a test set in order to give its current accuracy performance.

The DM takes extracted features and an array of "target labels". This array is set to the initially given labels (e.g. 10 labels) and the rest are NaN values, forming an array with same length as the extracted features dataframe. The idea is that this array will later include more labels as the oracle is queried for labels. The ALM is thereafter initialized.

Having initialized the DM and ALM, one can begin to run the loop. Some clarifications for following algorithm are (1) `dm.get_X_pool()` retrieves the extracted features imputed in the data manager, (2) `alm.query()` can take many instances if one seeks to identify many indexes, and (3) when using `alm.update()` one improves the active learning manager's ability to predict the labels since useful data points are queried.

Algorithm 1: Active Learning Loop

```
alm.fit(dm.get_X_pool())
accuracy = [ ]
for i in range(traning_instances - nbr_labels_given) do
    query_index = al_manager.query()
    query_label = training_labels[query_index]
    alm.update(query_index, query_label)
    label_prediction = alm.predict(test_features)
    prediction_accuracy = average(label_prediction == test_labels)
    accuracy.append(prediction_accuracy)
end
```

The predictions eventually stored in the list `accuracy` are saved and used for later comparison. Furthermore, the whole process is clocked in order to present duration of the process, also considered a key indicator.

5.3 Comparison

Firstly, it is intended to analyze which method perform best using uncertainty clustering sampling in regards to prediction accuracy: TSFEL, TSFRESH, or ACTS. It is deemed necessary to use uncertainty clustering sampling as the ACTS algorithm is slow. The methods will utilize a training set of 280 time series, label 200 of them during 8 iterations, therefore labeling 25 per iteration. Also, the ACTS algorithm have been implemented as described in Section 3.4. Acknowledged support has been presented, and my contribution to the implementation pf ACTS includes the Section 3.4.3.

Moreover, time duration for feature extraction using both TSFEL and TSFRESH will be analyzed. All features that can possibly be extracted using these libraries will be extracted. This can be done by clocking the time it takes using the python library `time`. Time duration for AL loop will be clocked for all methods.

The confidence intervals generated after having run 10 AL loops for TSFEL and TSFRESH will be analyzed. The confidence intervals will imply 95% confidence for a mean generated array from the 10 loops.The mean is an estimated mean for the 10 samples, using the standard deviation in order to construct a confidence interval For this process, more training data, 600 time series will be used. Also, uncertainty sampling will be used as AL method. It is in ones interest to compute the confidence intervals to see whether the results eventually converge, also if they reach better accuracy using more data.

With a possibility of extending the results section if satisfying results are obtained, it is possible to compare one of TSFEL and TSFRESH with random- and diversity sampling. It is reasoned that one method is only necessary to utilize for this comparison. With diversity sampling, uncertainty clustering sampling is implied. What is sought is to demonstrate when random vs. uncertainty sampling reaches convergence, and how much faster uncertainty clustering sampling can be and what implications that has.

Summarizing the comparison section, details are found in the table below:

Table 1: Training instances for each comparison carried out. Number of test instances depends on the dataset used

Amount Data	**ACTS Comparison**	**Confidence Intervals**	**Diversity & random sampling**
Training Instances	280	600	240

For ACTS implementation, see https://github.com/Willinki/ACTS

6 Results

Following tests have been carried out on an ASUS Laptop 18", Intel Core i7-6500U, 2 cores, 8GB RAM.

6.1 Duration

Table 2 shows duration for feature extraction for specified method and dataset

Table 2: Duration feature extraction for method and dataset, with size of dataset included

	Size (Nbr Time Series)	**TSFEL** (Seconds)	**TSFRESH** (Seconds)
Ball Bearing	5077	218	403
Production	2000	95	245
Swedish Leaf	1125	50	104

There is a linear relationship for the duration of the methods used and the size of the time series dataset. Further duration table includes time for ACTS, TSFEL, and TSFRESH to run through 1 loop using a training size of 250 time series, and labeling 200 with 8 iterations utilizing uncertainty clustering sampling.

Table 3: Duration AL loop for specified method and dataset

	TSFEL (Seconds)	**TSFRESH** (Seconds)	**ACTS** (Seconds)
Ball Bearing	5	10	1400
Production	4-6	10-12	1800
Swedish Leaf	3-5	10	1400

Moreover, adding duration for feature extraction with duration for AL loop one retrieves results presented in Table 4. There, it is also compared with the duration of ACTS.

Table 4: Duration AL loop for specified method and dataset

	TSFEL (Seconds)	**TSFRESH** (Seconds)	**ACTS** (Seconds)
Ball Bearing	225	415	1400
Production	100	260	1800
Swedish Leaf	55	60	1400

It should be noted that the majority of the duration presented for TSFEL and TSFRESH in Table 4 is due to feature extraction and that the AL loop is short. Feature extraction is a one time cost. Lastly, in Table 5 duration for uncertainty clustering sampling versus uncertainty sampling is shown, nevertheless only utilizing TSFRESH and a training size of 280 time series. Uncertainty clustering sampling is in the table referred to as diversity sampling.

Table 5: Duration for diversity- and uncertainty sampling for TSFRESH and specified dataset

	Diversity Sampling (Seconds)	**Uncertainty Sampling** (Seconds)
Ball Bearing	25	238
Production	23	303
Swedish Leaf	19	326

6.2 Accuracy ACTS vs. TSFEL and TSFRESH

Accuracies for ACTS, TSFEL, and TSFRESH are shown in figures and table below. Also, final accuracies shown in figures are means of five generated accuracies for each method.

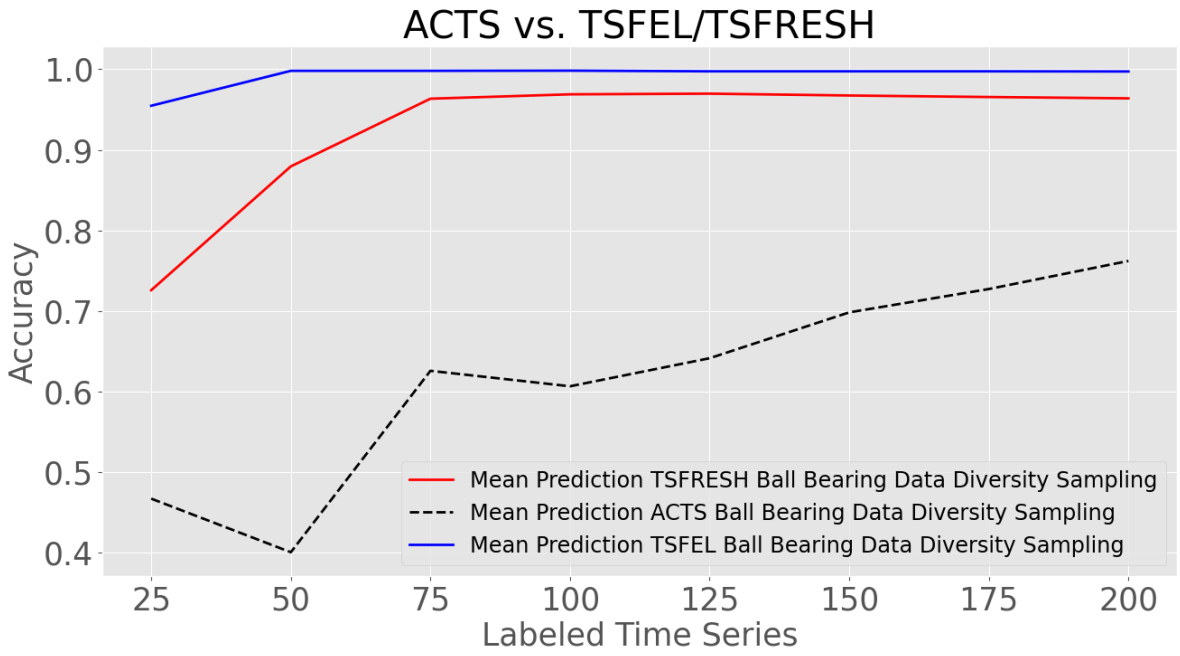

Figure 12: Mean predictions - ball bearing dataset. In total 8 iterations have been carried out for which 25 time series are labeled for one iteration

Approximately 280 (281) instances account for 5.54% of the ball bearing dataset. Therefore, 94.46% is used for testing. Final accuracy for TSFEL is 99.73 %. For TSFRESH, final accuracy is 96.39%. ACTS has a final accuracy of 76.21%.

For the production dataset 14.1% of the data is used for training, test size is consequently 85.9%. Final accuracy for TSFEL method is 79.28%. TSFRESH final accuracy is 79.74%. Final accuracy for ACTS is 75.55%.

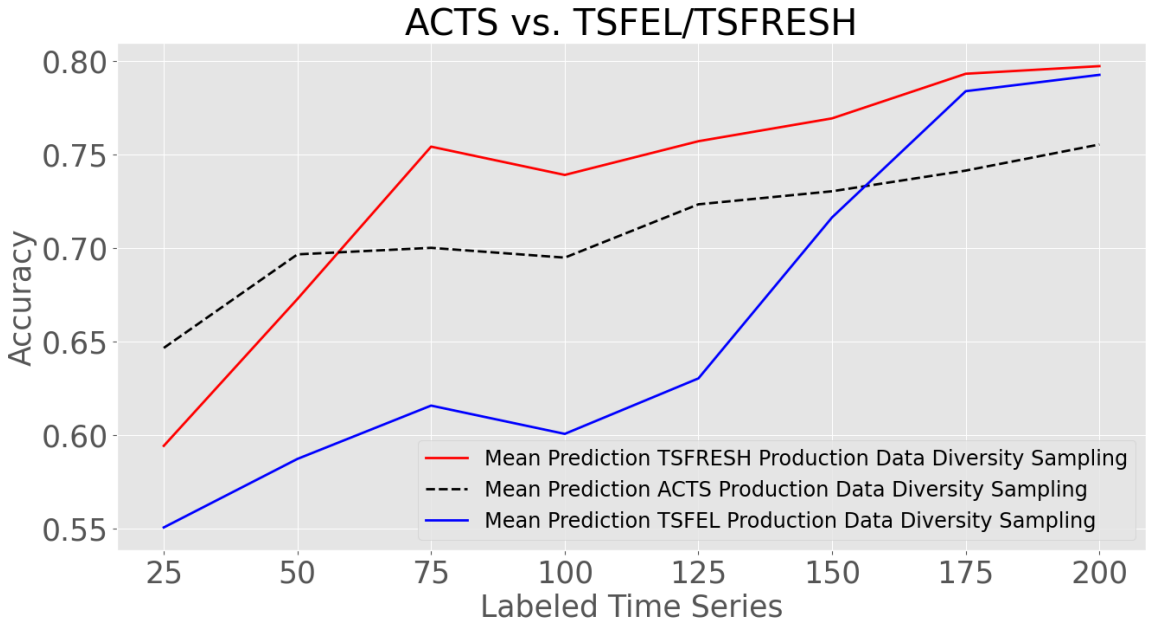

Figure 13: Mean predictions - production dataset. In total 8 iterations have been carried out for which 25 time series are labeled for one iteration

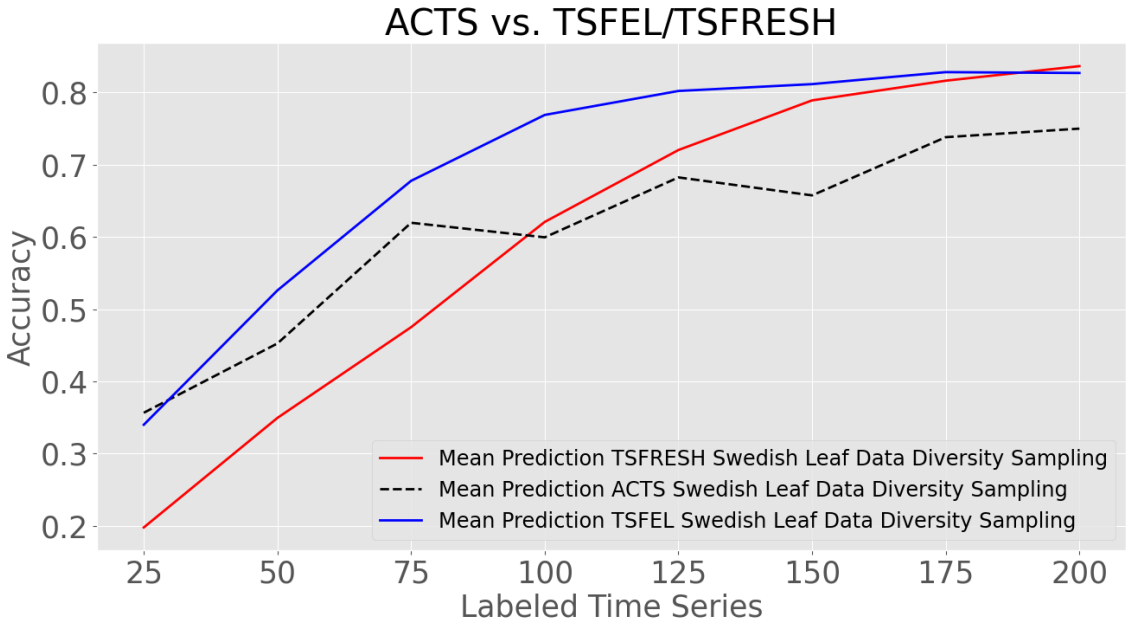

Figure 14: Mean predictions - Swedish leaf dataset. In total 8 iterations have been carried out for which 25 time series are labeled for one iteration

For the Swedish leaf dataset 25% of the data is used for training, test size is consequently 75%. Final accuracy for TSFEL method is 82.70%. TSFRESH final accuracy is 83.65%. Final accuracy for ACTS is 75.00%. See table below for summary of accuracy.

Table 6: Accuracy values for all dataset using all methods

Value	**TSFEL**	**TSFRESH**	**ACTS**
Accuracy Ball Bearing	99.73%	96.39%	76.21%
Accuracy Production	79.28%	79.74%	75.55%
Accuracy Swedish Leaf	82.70%	83.65%	75.00%

6.3 Confidence Intervals

Further result include confidence intervals and final accuracies for TSFEL and TSFRESH. The plots for TSFRESH can be found in the appendix. 95% confidence intervals after having generated 10 prediction series with 600 time series as training are shown in figures below. Note that 10 labels are given and therefore only 590 steps.

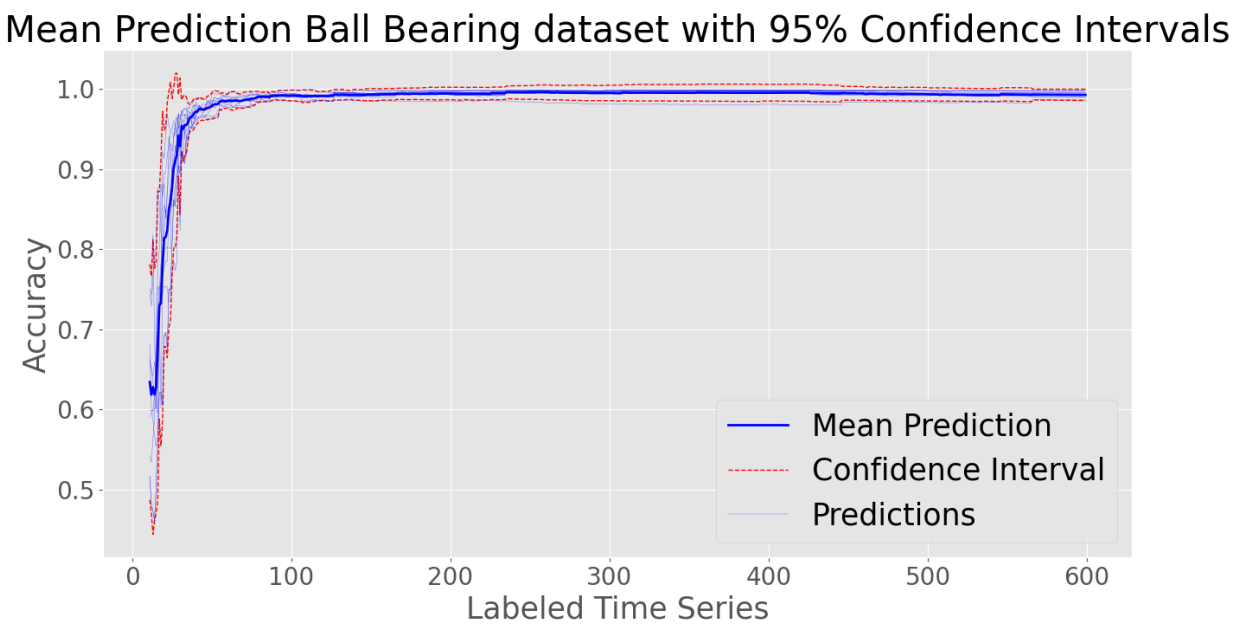

Figure 15: Confidence intervals generated for 10 AL loops using TSFEL and uncertainty sampling

Final accuracy for TSFEL on the ball bearing dataset is 0.993. For TSFRESH final accuracy achieved is 0.975.

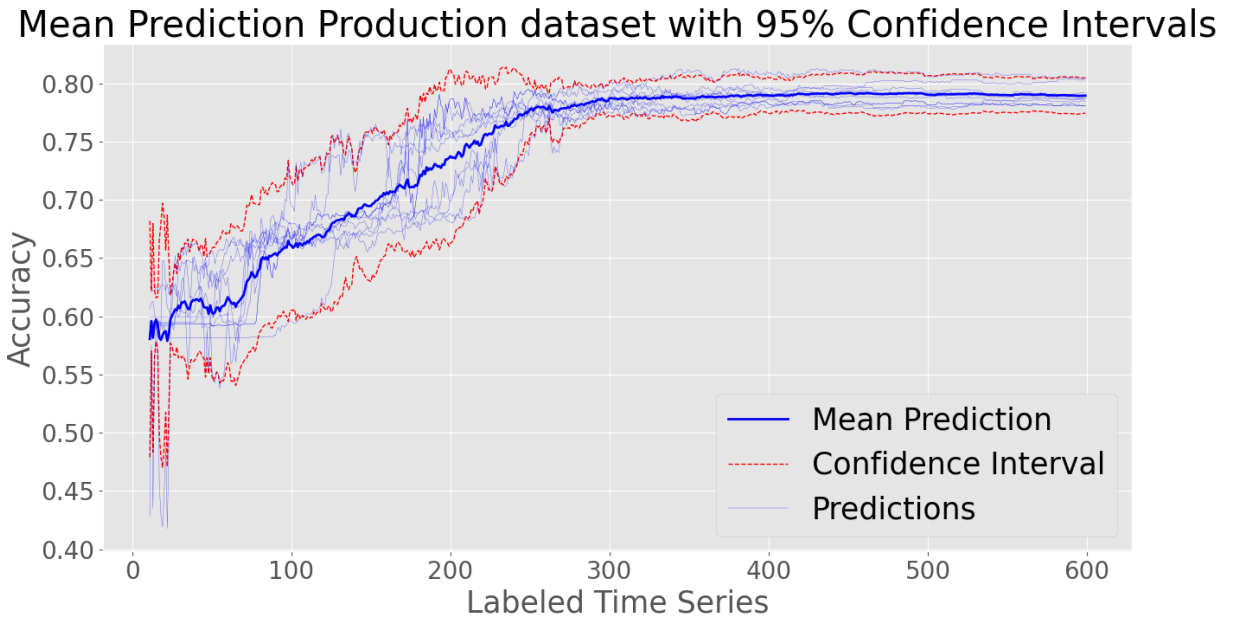

Figure 16: Confidence intervals generated for 10 AL loops using TSFEL and uncertainty sampling

For the production dataset final accuracy achieved for TSFEL is 0.785 and for TSFRESH it is 0.825. For TSFEL on the Swedish Leaf dataset using 575 time series for training, final accuracy reaches 0.898 and plot with confidence intervals can be found below

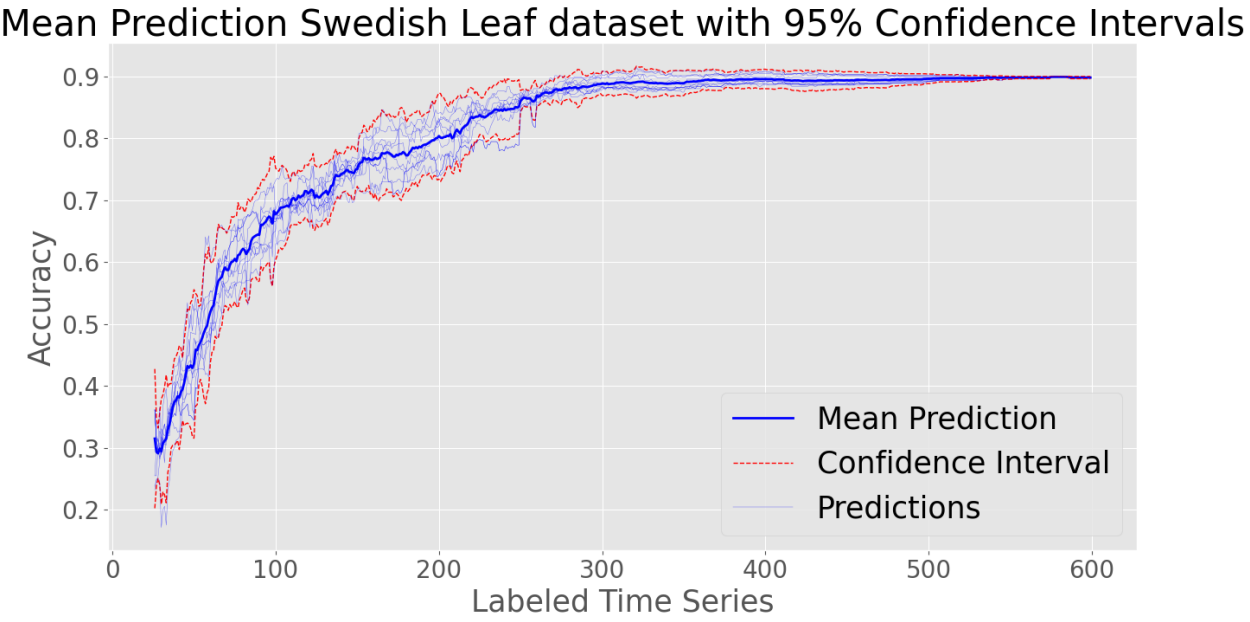

Figure 17: Confidence intervals generated for 10 AL loops using TSFEL and uncertainty sampling

The respective final accuracy for TSFRESH is 0.901. Plots for confidence intervals for TSFRESH can be found in appendix. Below in Table 7 one can identify final accuracies for the previous longer series generated with confidence intervals

Table 7: Final accuracies for series with 590 training time series using both TSFEL and TSFRESH and uncertainty sampling

	TSFEL (Final Accuracy)	**TSFRESH** (Final Accuracy)
Ball Bearing	0.993	0.976
Production	0.785	0.825
Swedish Leaf	0.898	0.884

6.4 Different Active Learning Methods

Moreover, the figures shown below include TSFRESH performance utilizing uncertainty sampling versus either random- or diversified sampling. The accuracy obtained for uncertainty-, random-, and diversity sampling is a mean of 10 loops. Figure 18 visualizes that the AL approach reaches convergence before method using random sampling. This is also true for the other two datasets but not as obvious, see Figure 23 and 25 in appendix. Diversity sampling achieves almost the same final accuracy, nevertheless it is somewhat lower than uncertainty sampling for each figure, see Figure 19, and 24 and 26 in appendix.

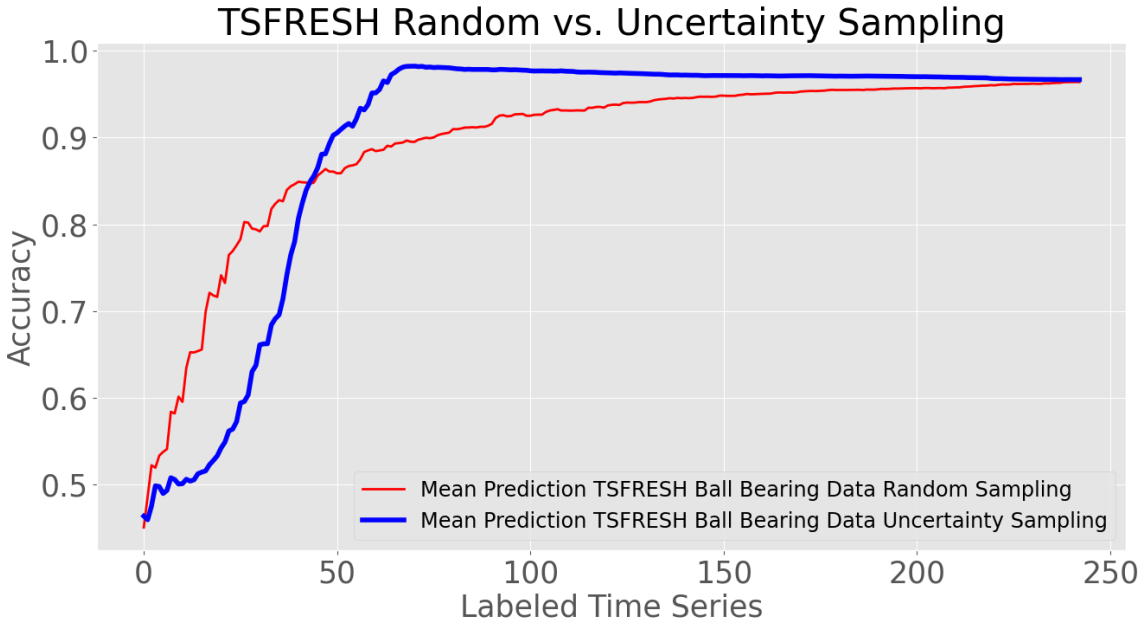

Figure 18: Random sampling vs. uncertainty sampling on ball bearing dataset using TSFRESH

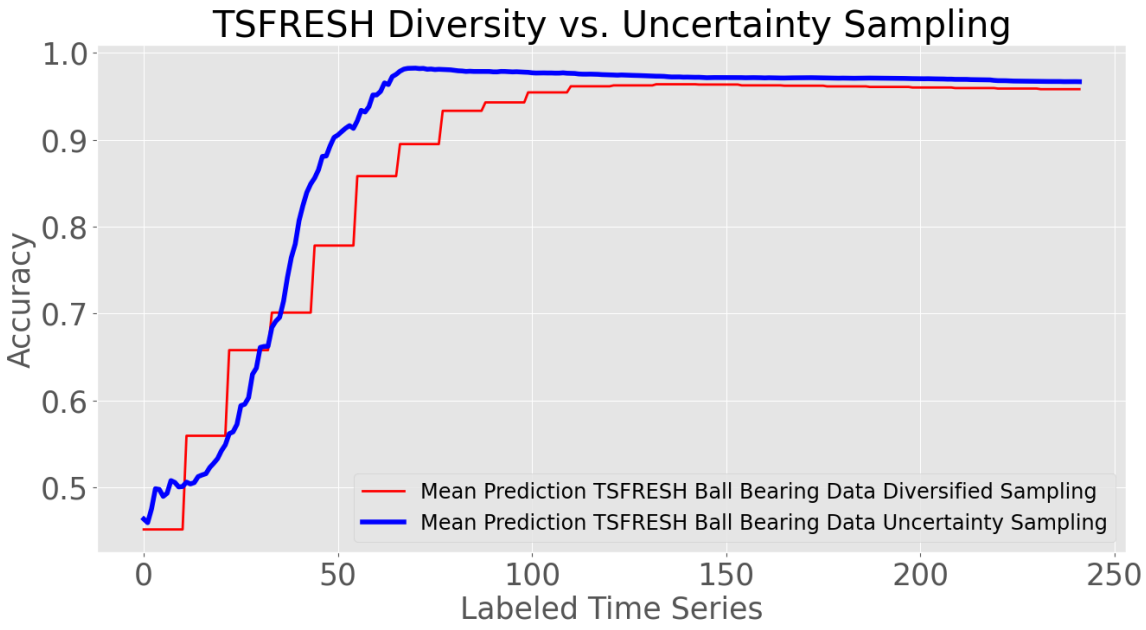

Figure 19: Diversity sampling vs. uncertainty sampling on ball bearing dataset using TSFRESH

7 Discussion

In current section, the results will be discussed, also including what it entails, questions that have arisen, and potential future implications. Firstly, the method comparison with the benchmark method ACTS will be discussed as well as the actual performances of the TSFEL/TSFRESH methods and what it implies. All performances on the three different datasets (1) benchmark, (2) uniform, and (3) Swedish leaf will be considered. Random- and diversity sampling will thereafter be discussed. Lastly, remaining questions and thesis contribution will be presented and summarized.

7.1 Comparison with ACTS

The comparison with ACTS using TSFEL and TSFRESH is presented initially in Section 6. Prior to testing, it is to be said that no assumptions were taken regarding the outcome, therefore the final results cannot be compared with earlier predictions, since they are non-existent. Nevertheless, it was noted when implementing ACTS, using steps in 3.4, that it would most certainly take long to run. Also, it is observed that the production dataset takes longer than ball bearing dataset for ACTS to process even though the dataset is larger. This is probably due to that the production dataset contains time series that longer than for other datasets. It is identified in Figures 12, 13, and 14 that both suggested methods outperforms ACTS. Stated is that the predictions are sample means of 5 loops, and one can clearly see that the accuracy for both TSFEL and TSFRESH average approximately 5% unit higher final accuracy. This is interesting as one could have expected ACTS to perform better. On the other hand, ACTS's accuracy obtained for the Swedish leaf dataset is in line with the results presented in (Peng et al. 2017). One might therefore argue that the implementation of this state-of-the-art method is correct. It can also be noted that for each dataset, ACTS's increase in accuracy is relatively low compared with the other two methods. It is difficult to explain this phenomena, but one may argue that it never reaches a convergence point but is still increasing at the end of all provided data points. Given more time series to be labeled, it is uncertain whether ACTS's accuracy would increase. However, looking at the relevant plots, one might argue that it would. Consequently, the accuracy of ACTS could reach the same level as for TSFEL and TSFRESH for datasets included. In Peng et al. (2017), they illustrate how ACTS reaches an error rate (inverse accuracy) of less than 20%. It is to be said that training size is much larger than what has been provided for this thesis. Given a larger training size, ACTS might had outperformed TSFEL and TSFRESH.

As stated, ACTS might have outperformed TSFEL and TSFRESH in terms of accuracy, nevertheless, duration of ACTS is enormous compared with the other methods. This is demonstrated in Table 2, 3, and 4. In matter of seconds, the feature extraction methods are able to give satisfying results. Nevertheless, some initial time in order to carry out feature extraction is necessary but when this is achieved, the loops only last for seconds. For ACTS, the duration is in hours. This increase in time due to extracting features is significant but still much shorter than for ACTS. This is arguably a huge advantage as speed could be considered a key factor in order to provide real-time solutions.

7.2 TSFEL vs. TSFRESH

It has been shown that both TSFEL and TSFRESH outperform ACTS which is satisfying. Nevertheless, both suggested methods include various number of features extracted. Comparing both methods for all datasets, one could have assumed that TSFRESH would have performed better due to more features extracted. This is not always the case, see comparison for benchmark dataset in Section 7.2.1. It is also noted from Table 4 that TSFEL in two cases only need half the time to extract features and complete an AL loop than for TSFRESH. Moreover, it can be identified in figures including confidence intervals that the results are fairly similar using 600 training instances instead of 280. Therefore, an increase of more than 300 time series does not improve the results dramatically, except for Swedish Leaf dataset. One should consider that 600 training instances are more than half of total amount of time series in the Swedish Leaf dataset (1125). Nevertheless, for uniform dataset it actually decreases. It is not obvious why this is the case but seen in Figure 16, there are some flaws with the labels provided for this dataset. This is further discussed in Section 7.2.2.

7.2.1 Benchmark Dataset Comparison

Discussing the performance on the different datasets, one can immediately see that extracting features using TSFEL and TSFRESH results is best applied for the benchmark dataset. Both methods reach accuracies of more than 95%, 99.73% and 96.39% respectively. It is possible that this is due to the characteristics of the different classes. In Figure 8 it is identified that they express different characteristics, some curves are not as

frequent or large. Nevertheless, it is interesting that by using TSFEL, therefore extracting 60 features, one is able to reach almost perfect accuracy. This is an argument that with a proper selection of features, one might accomplish greater performance in less time, depending on what dataset is used. There is an identified difference in total duration for both methods. Also, TSFEL reaches convergence point after having labeled 50 time series , compared with TSFRESH where convergence is reached after approx. 75 labeled time series. Lastly, it should be said that both methods use a small amount of training data, 5.54%, but is still able to achieve great accuracy. Interestingly, the achieved accuracy utilizing more training instances does not give more than marginal increase in accuracy (see Table 7). Actually, it decreases for TSFEL.

7.2.2 Uniform Dataset Comparison

Compared with the benchmark dataset, the achieved accuracies for both methods are poorer than for the benchmark dataset. There are several explanations to this. Firstly, it is already stated that the process of labeling the relevant time series was carried out by myself and Davide Badalotti, another thesis student at Viking Analytics. The fact that we are both not experts in labeling these types of time series might have affected the outcome of the labelling process. We did not complete the labelling simultaneously, nor were we communicating with each other while labelling was carried out. Therefore, there is a risk that labels are not 100% consistent due to that we determined individually which time series belonged to what class. Secondly, it is not always the case that one time series clearly includes a bump or no bump. Some cases were bordering to either one of them. False labelling could have occurred for these data points and model accuracy would consequently decrease. Considering possible "mislabellings", both methods still perform quite well. They are able to classify approximately 80% of the data points correctly after only being given 14.1% training data. Interestingly, TSFRESH performs better than TSFEL, however this is only marginal. In Table 7 it can be seen that TSFEL actually performs worse when utilizing 600 time series for training.

7.2.3 Swedish Leaf Comparison

Due to the number of classes and already having seen ACTS's performance on this dataset, it was assumed that both methods would not accomplish results as for the benchmark data set. However, they both demonstrate good classification, TSFRESH marginally better once more, achieving a final accuracy of 83.65%, compared with 82.70% for TSFEL. It is noted that for TSFRESH it seems that convergence is yet to be reached. Further testing with confidence intervals shows that convergence is achieved with more training instances. It is shown to be true in Table 7. Both TSFEL and TSFRESH reach approximately 90% which is a dramatic increase in terms of accuracy.

7.3 Random- and Diversity Sampling

From Figures 18, 23 and 25 it is evident that the active learning approaches perform better than random sampling. The difference is most clear for the benchmark dataset. This was expected as the core of active learning is to learn quickly. The "ramp-up" phase for uncertainty sampling in Figure 18 is unexpected. It is not clear why this occurs. One explanation might be the amount of labels given to the data manager. It is possible that with a greater initial set of given labels, one might see a steady rising curve that eventually converges, instead of a more "S-shaped"-like curve. Uncertainty sampling always seeks to exploit areas where it is not certain. If it therefore does not have significant amount of diversity in its training data, it might miss areas that are unknown to be uncertain. For the other datasets, the difference between random and uncertainty sampling is not as big. One reason could be that the training set is quite small.

For diversity sampling, it is possible in Figures 19, 24 and 26 to spot that final accuracy is not equal to the ones of uncertainty sampling. This is expected as there is a trade-off, performance vs. time. The final accuracies are almost the same as they have trained on the same data, however with the major difference being time duration. The time it takes to complete a run of uncertainty sampling is almost 10 times greater than the one for diversity sampling. One should still consider that uncertainty sampling "only" takes approximately 4 minutes. Nevertheless, it does not capture the time it takes for a human to label the time series.

7.4 Questions

Having discussed the results presented in Section 6, questions have arisen. It is found for all datasets that using a bigger set of extracted features does not necessarily improve performance, and if so, only marginal. Questions that arises through this line of argument are *what* set of features are optimal, and *how* may one obtain it? How

can proper feature selection be carried out? (Christ et al. 2018) introduce a feature selection approach which could be investigated. The "ramp-up" period identified in Figure 18 is worth investigating. Does it change with more given labels?

Furthermore, it does seem odd that final accuracy would decrease with more provided time series for training the classifier. It might therefore be interesting to have an individual with domain expertise label that uniform dataset and do the tests again.

Lastly, it is not certain how much faster the suggested method is when taking into consideration that manual labeling for AL is not included in duration for TSFEL and TSFRESH.

8 Conclusion and Future Research

Thesis includes the implementation of a state-of-the-art method (Peng et al. 2017) for time series active learning. Implementing this code has taken much of the time but it was crucial as it is used as a benchmark against suggested methods. Overall, time has mainly been spent on implementing algorithms and processes, amounting to 70-80% of overall time spent on thesis.

This thesis has attempted to identify a better method for time series active learning than a state-of-the-art method. It is found that suggested methods achieve better accuracy using presented datasets, and are also significatly faster. The datasets presented in Section 4 are reasoned to include a good degree of generality due to their varying size and number of classes. This implies that they may be used in every-day processes to extract information from time series data generated. Carrying out suggested method could potentially be done in real time and therefore have huge implications for the industry as one. Allowing for early detection of faults, uncertainties, or undesirable events could cut maintenance costs drastically. Also, it ensures a safer supply chain, and a transition towards Industry 4.0.

The suggested methods also come with the flexibility of diversified sampling, allowing for quicker processing. Considering a complex e.g. deep neural network that must be trained, one could utilize diversified sampling if training exceeds time it takes to label data points. Uncertainty sampling could be considered to be the main AL method used, while random sampling has acted as a benchmark.

Furthermore, suggested methods allow for a reasonably intuitive process to be carried out in an industrial setting. One can understand what features are being extracted and how they play a part in determining the class of a time series. Having used three datasets, one might argue that they add to robustness as they include a variety of number of classes. Moreover, one dataset is common for this thesis and the state-of-the-art method paper. TSFEL is in this thesis arguably better as it achieves similar or better accuracy as TSFRESH. It computes approximately half the amount of features as for TSFRESH and is faster to process. The features computed are all included in the both packages.

Future research should attempt to optimize a feature selection process to derive an optimal set of features for the data collected. This consequently could allow for greater interpretability and understanding of predictive maintenance in the industry. Moreover, clocking the physical labelling carried out by a person with domain expertise could give this work an extra dimension. This would allow for real-world cases and to compare the actual total time it takes to extract and label the time series. Also, this thesis could most certainly be extended so that it includes a greater variety of datasets, perhaps including all datasets introduced in the state-of-the-art (Peng et al. 2017) method paper, or more from the UCR dataset library (Chen et al. 2015). This thesis could also be extended by including more active learning approaches e.g. query-by-committee. Automated feature extraction could also be combined image recognition approaches, perhaps resulting in a more robust predictive maintenance process. Lastly, it would be interesting to test the ACTS method using more training instances on a better machine. One could there through derive more precise accuracies for more training data, that could be compared with feature extraction methods presented.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C. & Wang, Y. (2020), 'Gluonts: Probabilistic and neural time series modeling in python', *Journal of Machine Learning Research* **21**, 1–6.

Atzori, L., Iera, A. & Morabito, G. (2010), 'The Internet of Things: A survey', *Computer Networks* **54**(15), 2787–2805.

Bach, S. H., He, B., Ratner, A. & Ré, C. (2017), 'Learning the structure of generative models without labeled data', *34th International Conference on Machine Learning, ICML 2017* **1**, 434–449.

Bagnall, A., Lines, J., Bostrom, A., Large, J. & Keogh, E. (2017), 'The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances', *Data mining and knowledge discovery* **31**(3), 606–660.

Bajic, B., Cosic, I., Lazarevic, M., Sremcev, N. & Rikalovic, A. (2018), 'Machine learning techniques for smart manufacturing: Applications and challenges in industry 4.0', *Department of Industrial Engineering and Management Novi Sad, Serbia* **29**.

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T. & Gamboa, H. (2020), 'TSFEL: Time Series Feature Extraction Library', *SoftwareX* **11**, 100456.
URL: *https://doi.org/10.1016/j.softx.2020.100456*

Bellman, R. (1957), *Dynamic Programming*, Princeton University Press.

Bertolini, M., Mezzogori, D., Neroni, M. & Zammori, F. (2021), 'Machine Learning for industrial applications: a comprehensive literature review', *Expert Systems with Applications* **175**(December 2020), 114820.
URL: *https://doi.org/10.1016/j.eswa.2021.114820*

Bevilacqua, M. & Braglia, M. (2000), 'Analytic hierarchy process applied to maintenance strategy selection', *Reliability Engineering and System Safety* **70**(1), 71–83.

Bhardwaj, P. (2019), 'Types of sampling in research', *Journal of the Practice of Cardiovascular Sciences* **5**(3), 157.

Carreira-Perpinán, M. (1997), 'A review of dimension reduction techniques', *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09* pp. 1–69.
URL: *http://www.pca.narod.ru/DimensionReductionBrifReview.pdf*

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A. & Batista, G. (2015), 'The ucr time series classification archive'. www.cs.ucr.edu/~eamonn/time_series_data/.

Christ, M., Braun, N., Neuffer, J. & Kempa-Liehr, A. W. (2018), 'Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)', *Neurocomputing* **307**, 72–77.
URL: *https://www.sciencedirect.com/science/article/pii/S0925231218304843*

Cohn, D. A., Ghahramani, Z. & Jordan, M. I. (1996), 'Active learning with statistical models', *Journal of artificial intelligence research* **4**, 129–145.

Cortes, C. & Vapnik, V. (1995), 'Support vector networks', *Machine Learning* **20**, 273–297.

Cunningham, P. & Delany, S. (2007), 'k-nearest neighbour classifiers', *Mult Classif Syst* .

Dachyar, M., Zagloel, T. Y. M. & Saragih, L. R. (2019), 'Knowledge growth and development: internet of things (IoT) research, 2006–2018', *Heliyon* **5**(8), e02264.
URL: *https://doi.org/10.1016/j.heliyon.2019.e02264*

Danka, T. & Horvath, P. (2018), 'modAL: A modular active learning framework for Python'. available on arXiv at https://arxiv.org/abs/1805.00979.
URL: *https://github.com/cosmic-cortex/modAL*

De Mauro, A., Greco, M. & Grimaldi, M. (2016), 'A formal definition of Big Data based on its essential features', *Library Review* **65**(3), 122–135.

Dennis, D. K., Acar, D. A. E., Mandikal, V., Sadasivan, V. S., Simhadri, H. V., Saligrama, V. & Jain, P. (2019), 'Shallow RNNs: A method for accurate time-series classification on tiny devices', *Advances in Neural Information Processing Systems* **32**(NeurIPS), 1–15.

Dorst, W., Glohr, C., Hahn, T., Knafla, F., Loewen, U., Rosen, R. & Sandner, M. (2016), *Implementation strategy Industrie 4.0: Report on the results of the Industrie 4.0 Platform*, number January.
URL: *https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2016/januar/Implementation_Strat_Report_on_the_results_of_Industrie_4.0_Platform/Implementation-Strategy-Industrie-40-ENG.pdf*

Elmasri, R. & Lee, J. Y. (1998), *Implementation options for time-series data*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 115–128.
URL: *https://doi.org/10.1007/BFb0053700*

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L. & Muller, P. A. (2019), 'Transfer learning for time series classification', *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018* pp. 1367–1376.

Franc, V. & Hlaváč, V. (2002), 'Vector machine multi-class support', *Proceedings - International Conference on Pattern Recognition* **2**(October 2014), 236–239.

Fulcher, B. D. & Jones, N. S. (2014), 'Highly comparative feature-based time-series classification', *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Fulcher, B. D. & Jones, N. S. (2017), 'hctsa: A Computational Framework for Automated Time-Series Phenotyping Using Massive Feature Extraction', *Cell Systems* **5**(5), 527–531.e3.
URL: *https://doi.org/10.1016/j.cels.2017.10.001*

Ge, L. & Ge, L.-J. (2016), 'Feature extraction of time series classification based on multi-method integration', *Optik* **127**(23), 11070–11074.
URL: *https://www.sciencedirect.com/science/article/pii/S0030402616309652*

Gocheva-Ilieva, S., Voynikova, D., Stoimenova, M., Ivanov, A. & Iliev, I. (2019), 'Regression trees modeling of time series for air pollution analysis and forecasting', *Neural Computing and Applications* **31**.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. http://www.deeplearningbook.org.

He, G., Duan, Y., Li, Y., Qian, T., He, J. & Jia, X. (2016), 'Active learning for multivariate time series classification with positive unlabeled data', *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI* **2016-Janua**, 178–185.

Ilhan, H. O. & Amasyalı, M. F. (2014), 'Active Learning as a Way of Increasing Accuracy', *International Journal of Computer Theory and Engineering* **6**(6), 460–465.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P. A. (2019), 'Deep learning for time series classification: a review', *Data Mining and Knowledge Discovery* **33**(4), 917–963.

Karim, F., Majumdar, S. & Darabi, H. (2019), 'Insights into lstm fully convolutional networks for time series classification', *IEEE Access* **7**, 67718–67725.

Kobylin, O. & Lyashenko, V. (2020), 'Time series clustering based on the k-means algorithm', *Journal La Multiapp* **1**(3), 1–7.
URL: *https://newinera.com/index.php/JournalLaMultiapp/article/view/191*

Konyushkova, K., Sznitman, R. & Fua, P. (2017), Learning active learning from data, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 30, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper/2017/file/8ca8da41fe1ebc8d3ca31dc14f5fc56c-Paper.pdf*

Krawczak, M. & Szkatuła, G. (2014), 'An approach to dimensionality reduction in time series', *Information Sciences* **260**, 15–36.

Lei, Y. & Wu, Z. (2020), 'Time series classification based on statistical features', *EURASIP Journal on Wireless Communications and Networking* **2020**(1), 1–13.

Lewis, D. D. & Catlett, J. (1994), *Heterogeneous Uncertainty Sampling for Supervised Learning*, Morgan Kaufmann Publishers, Inc.
URL: *http://dx.doi.org/10.1016/B978-1-55860-335-6.50026-X*

Li, G., Hou, Y. & Wu, A. (2017), 'Fourth Industrial Revolution: technological drivers, impacts and coping methods', *Chinese Geographical Science* **27**(4), 626–637.

Li, Z., Wang, K. & He, Y. (2016), 'Industry 4.0-potentials for predictive maintenance', *Advances in Economics, Business and Management Research* pp. 1–5.

Lin, J., Keogh, E., Wei, L. & Lonardi, S. (2007), 'Experiencing SAX: A novel symbolic representation of time series', *Data Mining and Knowledge Discovery* **15**(2), 107–144.

Lines, J. & Bagnall, A. (2015), 'Time series classification with ensembles of elastic distance measures', *Data Mining and Knowledge Discovery* **29**(3), 565–592.

Lines, J., Taylor, S. & Bagnall, A. (2018), 'Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles', *ACM Trans. Knowl. Discov. Data* **12**(5).
URL: *https://doi.org/10.1145/3182382*

Liu, Y. & Xu, X. (2017), 'Industry 4.0 and cloud manufacturing: A comparative analysis', *Journal of Manufacturing Science and Engineering, Transactions of the ASME* **139**(3), 1–8.

Mierswa, I. & Morik, K. (2005), 'Automatic feature extraction for classifying audio data', *Machine learning* **58**(2-3), 127–149.

Mitchell, T. M. (1997), *Machine Learning*, 1 edn, McGraw-Hill, Inc., USA, pp. 20–52.

Monarch, R. (2021), *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*, Manning Publications.
URL: *https://books.google.se/books?id=LCh0zQEACAAJ*

Montero, A. G. & Blanc, D. L. (2019), Lessons for Today from Past Periods of Rapid Technological Change, Working Papers 158, United Nations, Department of Economics and Social Affairs.
URL: *https://ideas.repec.org/p/une/wpaper/158.html*

Mourtzis, D., Vlachou, E. & Milas, N. (2016), 'Industrial Big Data as a Result of IoT Adoption in Manufacturing', *Procedia CIRP* **55**, 290–295.
URL: *http://dx.doi.org/10.1016/j.procir.2016.07.038*

Nanopoulos, A., Alcock, R. & Manolopoulos, Y. (2001), 'Feature-based Classification of Time-series Data', *International Journal of Computer Research* **10**(3), 49–61.

Naul, B., van der Walt, S., Crellin-Quick, A., Bloom, J. & Pérez, F. (2016), 'cesium: Open-Source Platform for Time-Series Inference', *Proceedings of the 15th Python in Science Conference* (Scipy), 27–35.

Nun, I., Protopapas, P., Sim, B., Zhu, M., Dave, R., Castro, N. & Pichara, K. (2015), 'FATS: Feature Analysis for Time Series', pp. 1–13.
URL: *http://arxiv.org/abs/1506.00010*

O'Donovan, P., Leahy, K., Bruton, K. & O'Sullivan, D. T. (2015), 'An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities', *Journal of Big Data* **2**(1), 1–26.

Peng, F., Luo, Q. & Ni, L. M. (2017), Acts: an active learning method for time series classification, *in* '2017 IEEE 33rd International Conference on Data Engineering (ICDE)', IEEE, pp. 175–178.

Perrault, R., Shoham, Y., Brynjolfsson, E., Clark, J., Etchemendy, J., Grosz, B., Lyons, T., Manyika, J., Mishra, S. & Niebles, J. C. (2019), 'Artificial Intelligence Index 2019 Annual Report', *Human-Centered AI Institute, Stanford University* p. 291.
URL: *https://hai.stanford.edu/sites/g/files/sbiybj10986/f/ai_index_2019_report.pdf*

Radziwon, A., Bilberg, A., Bogers, M. & Madsen, E. S. (2014), 'The smart factory: Exploring adaptive and flexible manufacturing solutions', *Procedia Engineering* **69**, 1184–1190.
URL: *http://dx.doi.org/10.1016/j.proeng.2014.03.108*

Randall, R. B. & Antoni, J. (2011), 'Rolling element bearing diagnostics—a tutorial', *Mechanical Systems and Signal Processing* **25**(2), 485–520.
URL: *https://www.sciencedirect.com/science/article/pii/S0888327010002530*

Russell, S. & Norvig, P. (2009), *Artificial Intelligence: A Modern Approach*, 3rd edn, Prentice Hall Press, USA, pp. 1–87.

Russell, S. & Norvig, P. (2020), *Artificial Intelligence: A Modern Approach*, 4th edn, Prentice Hall Press, USA.

Sarker, I. H. (2021), 'Machine Learning: Algorithms, Real-World Applications and Research Directions', *SN Computer Science* **2**(3).
URL: *https://doi.org/10.1007/s42979-021-00592-x*

Sarma, A. C. & Girão, J. (2009), 'Identities in the future internet of things', *Wireless Personal Communications* **49**(3), 353–363.

Selcuk, S. (2017), 'Predictive maintenance, its implementation and latest trends', *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **231**(9), 1670–1679.

Settles, B. (2011), 'From Theories to Queries: Active Learning in Practice', *Proceedings of the Workshop on Active Learning and Experimental Design* **16**, 1–18.

Sezer, O. B., Gudelek, M. U. & Ozbayoglu, A. M. (2020), 'Financial time series forecasting with deep learning: A systematic literature review: 2005–2019', *Applied Soft Computing* **90**, 106–181.

Sharp, M., Ak, R. & Hedberg, T. (2018), 'A survey of the advancing use and development of machine learning in smart manufacturing', *Journal of Manufacturing Systems* **48**, 170–179.
URL: *https://doi.org/10.1016/j.jmsy.2018.02.004*

Smith, W. A. & Randall, R. B. (2015), 'Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study', *Mechanical Systems and Signal Processing* **64-65**, 100–131.
URL: *http://dx.doi.org/10.1016/j.ymssp.2015.04.021*

Söderkvist, O. (2001), Computer vision classification of leaves from swedish trees, number 3132 *in* 'LiTH-ISY-Ex', p. 74.

Tavakoli, N., Siami-Namini, S., Adl Khanghah, M., Mirza Soltani, F. & Siami Namin, A. (2020), An autoencoder-based deep learning approach for clustering time series data, Vol. 2. Funding Information: This work is partially funded by the support from National Science Foundation under the Grant Numbers 1564293, 1723765, and 1821560. Publisher Copyright: © 2020, Springer Nature Switzerland AG.

Tealab, A. (2018), 'Time series forecasting using artificial neural networks methodologies: A systematic review', *Future Computing and Informatics Journal* **3**(2), 334–340.
URL: *https://doi.org/10.1016/j.fcij.2018.10.003*

Turner, D. P. (2020), Sampling methods in research design, Wiley Online Library, pp. 8–12.

Tzotsos, A. & Argialas, D. (2008), Support vector machine classification for object-based image analysis, *in* 'Object-Based Image Analysis', Springer, pp. 663–677.

Urner, R., Wulff, S. & Ben-David, S. (2013), Plal: Cluster-based active learning, *in* S. Shalev-Shwartz & I. Steinwart, eds, 'Proceedings of the 26th Annual Conference on Learning Theory', Vol. 30 of *Proceedings of Machine Learning Research*, PMLR, Princeton, NJ, USA, pp. 376–397.
URL: *http://proceedings.mlr.press/v30/Urner13.html*

Vaidya, S., Ambad, P. & Bhosle, S. (2018), 'Industry 4.0 - A Glimpse', *Procedia Manufacturing* **20**, 233–238.
URL: *https://doi.org/10.1016/j.promfg.2018.02.034*

Van Der Maaten, L. J. P., Postma, E. O. & Van Den Herik, H. J. (2009), 'Dimensionality Reduction: A Comparative Review', *Journal of Machine Learning Research* **10**(June 2014), 1–41.

van Kuppevelt, D., Meijer, C., Huber, F., van der Ploeg, A., Georgievska, S. & van Hees, V. T. (2020), 'Mcfly: Automated deep learning on time series', *SoftwareX* **12**, 100548.
URL: *https://doi.org/10.1016/j.softx.2020.100548*

Van Rossum, G. & Drake Jr, F. L. (1995), *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam.

Vogel-Heuser, B. & Jumar, U. (2019), 'Scientific fundamentals of Industry 4.0', *At-Automatisierungstechnik* **67**(6), 502–503.

Witten, I., Frank, E. & Hall, M. (2011), *Data Mining - Practical Machine Learning Tools and Techniques, 3rd Edition*.

Wuest, T., Weimer, D., Irgens, C. & Thoben, K. D. (2016), 'Machine learning in manufacturing: Advantages, challenges, and applications', *Production and Manufacturing Research* **4**(1), 23–45.
URL: *http://dx.doi.org/10.1080/21693277.2016.1192517*

Xu, X. & Hua, Q. (2017), 'Industrial Big Data Analysis in Smart Factory: Current Status and Research Strategies', *IEEE Access* **5**(c), 17543–17551.

Yamato, Y., Fukumoto, Y. & Kumazaki, H. (2017), 'Predictive maintenance platform with sound stream analysis in edges', *Journal of Information Processing* **25**, 317–320.

Yan, J., Meng, Y., Lu, L. & Li, L. (2017), 'Industrial big data in an industry 4.0 environment: Challenges, schemes and applications for predictive maintenance', *IEEE Access* **5**, 1–8.

Ye, L. & Keogh, E. (2009), Time series shapelets: A new primitive for data mining, *in* 'Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '09, Association for Computing Machinery, New York, NY, USA, p. 947–956.
URL: *https://doi.org/10.1145/1557019.1557122*

Yen, G. G. (2000), 'Wavelet packet feature extraction for vibration monitoring', *IEEE Transactions on Industrial Electronics* **47**(3), 650–667.

9 Appendix

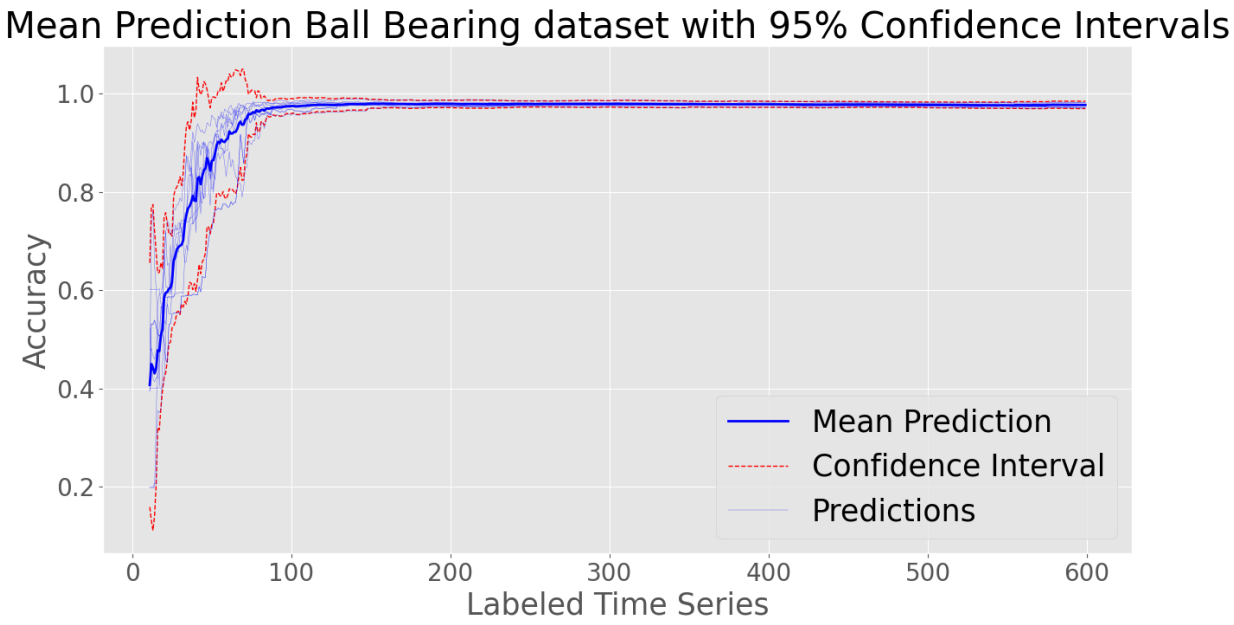

Figure 20: Confidence intervals generated for 10 AL loops using TSFRESH and uncertainty sampling

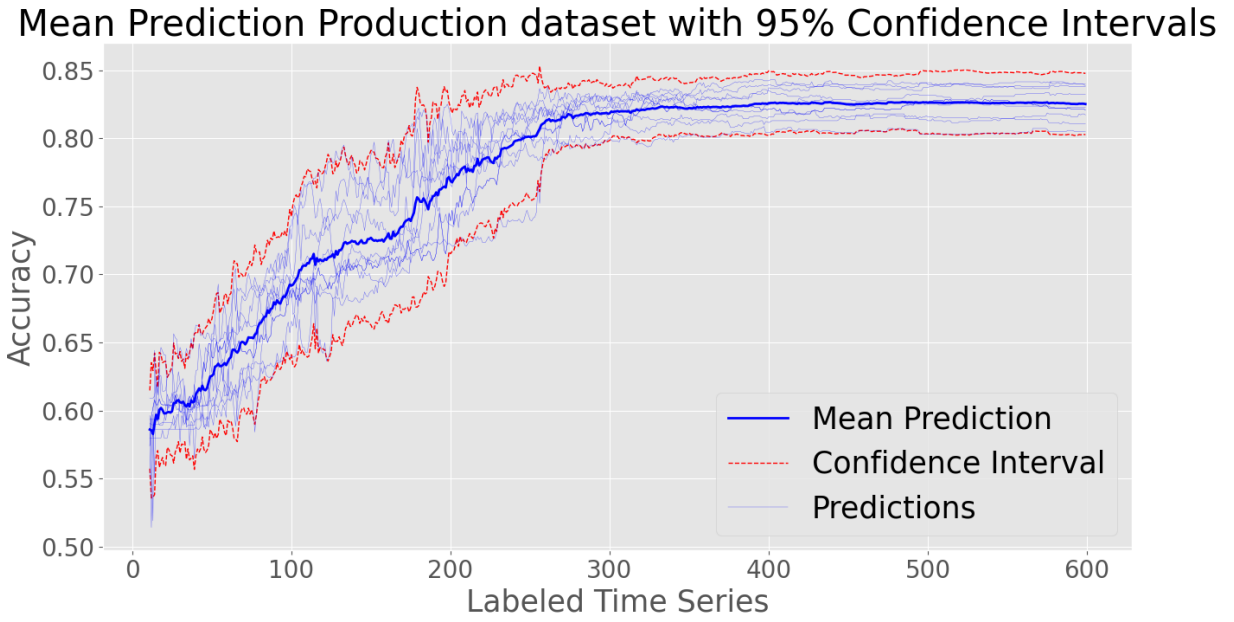

Figure 21: Confidence intervals generated for 10 AL loops using TSFRESH and uncertainty sampling

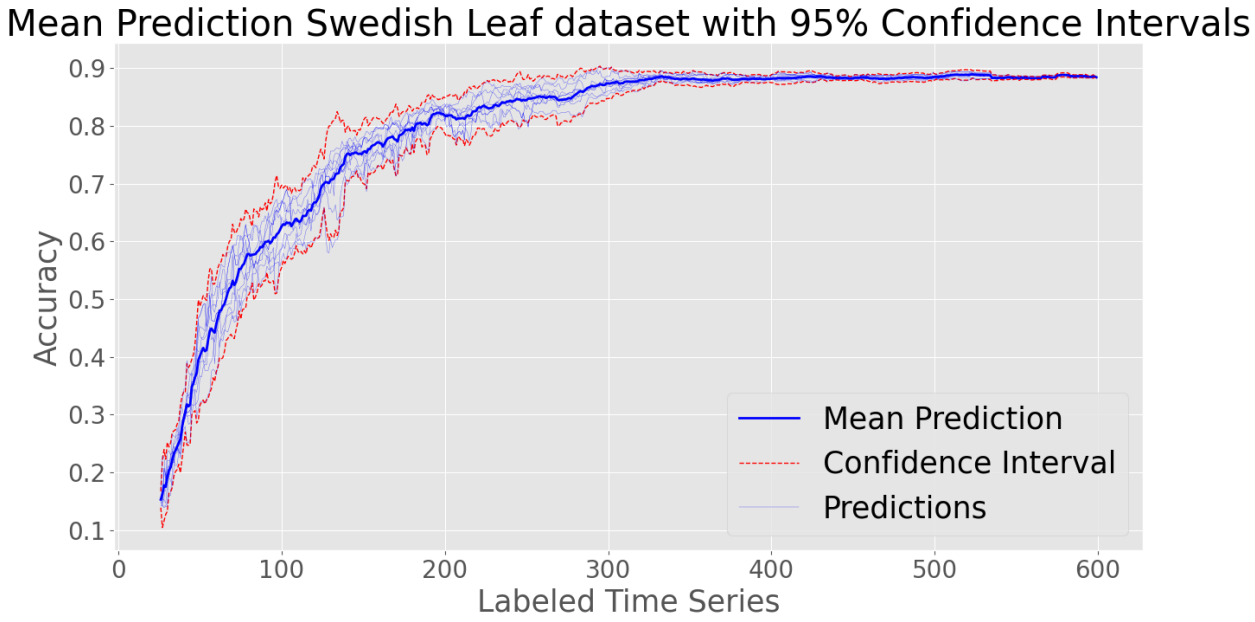

Figure 22: Confidence intervals generated for 10 AL loops using TSFRESH and uncertainty sampling

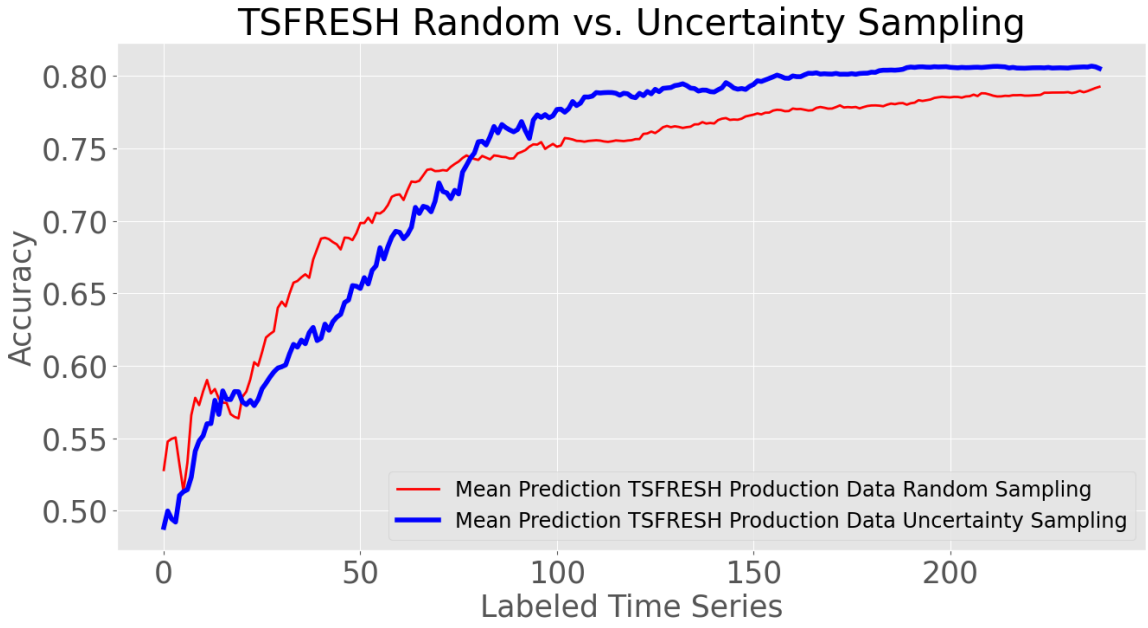

Figure 23: Random sampling vs. uncertainty sampling on production dataset using TSFRESH

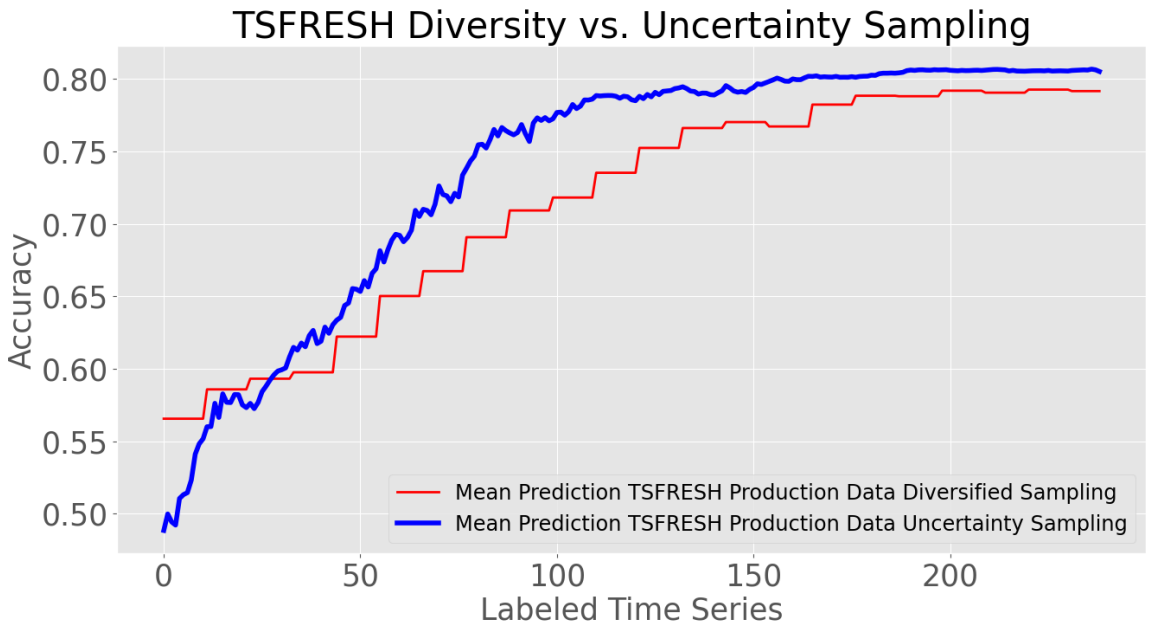

Figure 24: Diversity sampling vs. uncertainty sampling on production dataset using TSFRESH

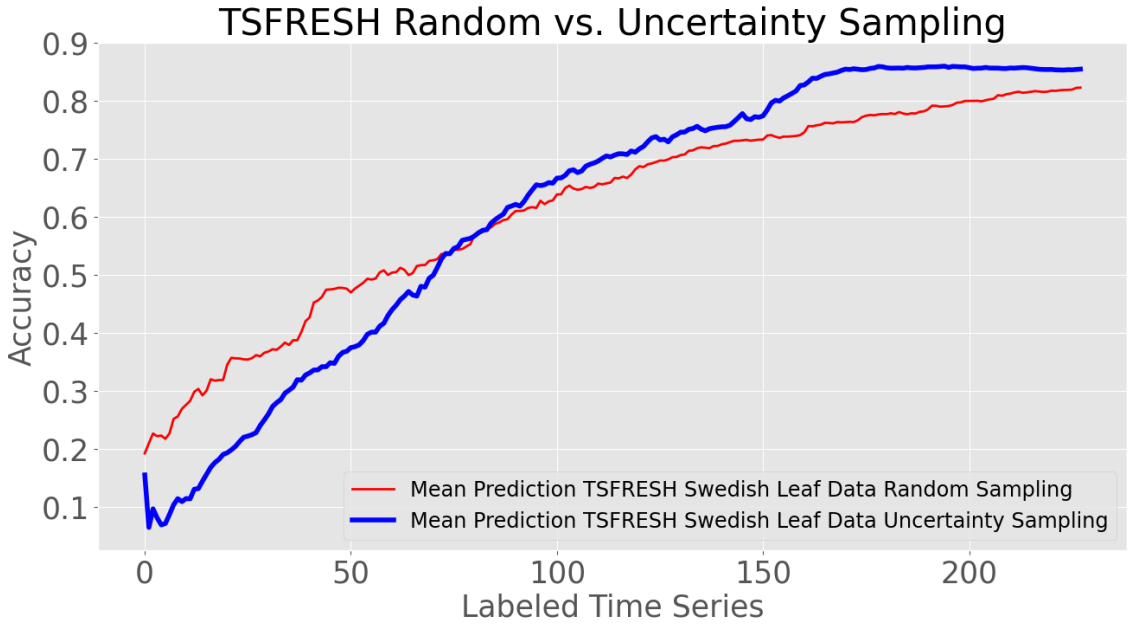

Figure 25: Random sampling vs. uncertainty sampling on Swedish leaf dataset using TSFRESH

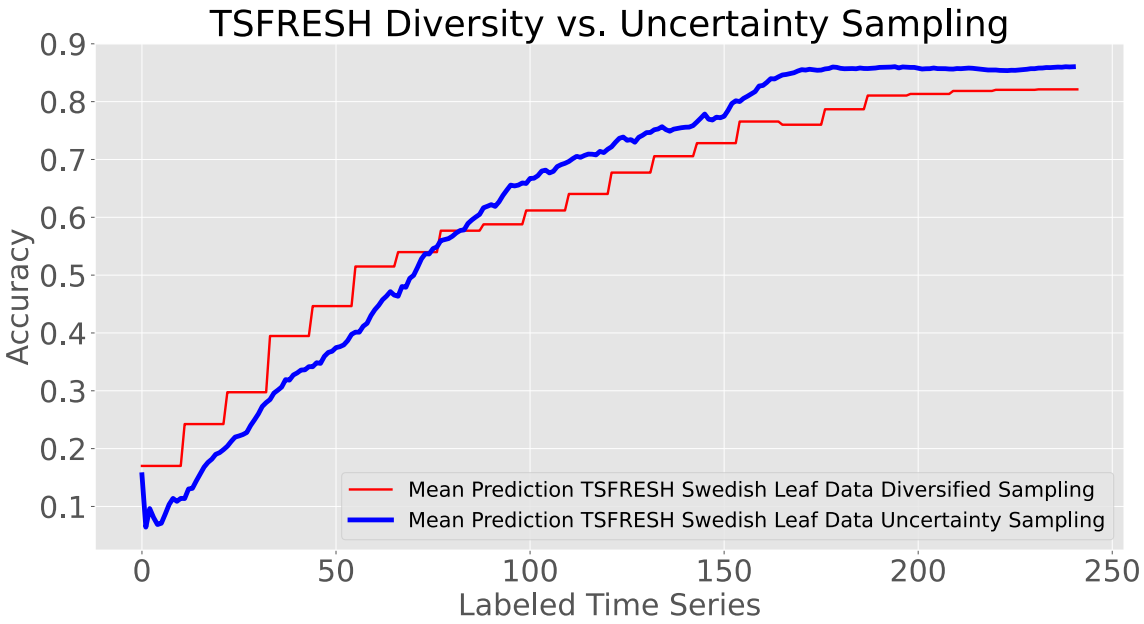

Figure 26: Diversity sampling vs. uncertainty sampling on Swedish Leaf dataset using TSFRESH