Ministry of Education and Science of the Russian Federation
Federal State Autonomous Educational Institution of Higher Education
«Ural Federal University named after the first President of Russia B. N. Yeltsin»

Engineering School of Information Technologies, Telecommunications and Control Systems

MASTER THESIS

Sport Activity Classification Using Classical Machine Learning and Time Series Methods

Supervisor: Anton Dolganov

Student of the group: RIM-210972

Name: Jarno Matarmaa

Yekaterinburg
2023

ABSTRACT

Human activity recognition has been well investigated field for a decade especially using inertial sensor data recorded by a smartphone or sport-watch, or any device which can be used daily such as activity trackers. However, this type of data is not always available, therefore the development of alternative methods may be reasonable. In this thesis, retrospective personalized supervised sport activity classification is implemented applying a comprehensive set of classic machine learning and time series classification algorithms into the sport data recorded by an individual athlete. Retrospective in the sense that the actual classification happens after completing the activity and finishing the recording. The data consists of five categories biking, running, walking, skiing, and roller skiing and extracted features from a several sensors such as heart rate, geolocation, barometer, and accelerometer, and by contextualizing sensor data with a personal data generating a calorie consumption feature, etc. The same original data is structured in three different ways, since there are three diverging types of classifiers, in order to find the best method to label sport activities. Classification tasks are divided into a standard classical machine learning, univariate time series classification and multivariate time series classification. In time series classification particular set of features is used, namely heart rate, speed, and altitude, and also classification task is reduced to three categories, biking, running and other. For univariate classification, this three-dimensional multivariate data is transformed into a vector using interlacing method, whereas applied multivariate classifier has the integrated ability to construct applicable data from features. The obtained results show that by using 21 relevant features, five categories were clearly separable producing over 95 percent accuracy by several standard machine learning models, such as support vector classifier, logistic regression classifiers, and k-nearest neighbours' classifier. Adopted multivariate model MUSE, which conducts word extraction from the features achieved the accuracy of 96.6 % but requiring an order of magnitude more time. Applied interlacing method for a univariate data transformation was also successful, since several models performed by over 90 % accuracy, and best of them, supervised time series forest classifier up to 93.1 %. Based on the analysis, all applied methods have their strengths and the method choice ultimately depends on the available data.

Keywords: machine learning, time series classification, univariate, multivariate signal, sport activity classification, interlaced signal, sklearn, sktime, supervised learning, sensor-based activity recognition, human activity recognition, multi-class classification

CONTENTS

ABBREVIATIONS

API	Application Programming Interface
AUC	Area Under Curve
CNN	Convolutional Neural Network
CSV	Comma Separated Values
DL	Deep Learning
GC	Garmin Connect
GPS	Global Positioning System
HAR	Human Activity Recognition
LSTM	Long-Short Term Memory
M-TSC	Multivariate Time Series Classification
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristics
S-CML	Standard Classical Machine Learning
SAC	Sport Activity Classification
SAR	Sport Activity Recognition
TCX	Training Center XML
TS-SAC	Time Series Sport Activity Classification
TSA	Time Series Analysis
TSC	Time Series Classification
U-TSC	Univariate Time Series Classification
XML	Extensible Markup Language
t-SNE	t-distributed Stochastic Neighbor Embedding
CBOSS	Contractable Bag of Symbolic Fourier Approximation Symbols
CTSF	Composable Time Series Forest
CE	Column Ensemble
DT	Decision Tree
G-NB	Gaussian Naïve Bayes
GB	Gradient Boosting
LDA	Linear Discriminant Analysis

LR	Logistic Regression
MLP	Multilayer Perceptron
MUSE	Multivariate Symbolic Extension
NB	Naïve Bayes Classifier
QDA	Quadratic Discriminant Analysis
RF	Random Forest
RIC	Random Interval Classifier
RISE	Random Interval Spectral Ensemble
STC	Shapelet Transform Classifier
STSF	Supervised Time Series Forest
SVM	Support Vector Machine
TSF	Time Series Forest
WEASEL	Word Extraction for Time Series Classification
kNN	k-Nearest Neighbors
kNN-TS	k-Nearest Neighbors Time Series

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

INTRODUCTION

1.1 Problem description

Nowadays sport watches are collecting and storing an enormous amount of data from a wide variety of sport activities. Sport watches, as well as many kinds of smart watches and smartphones which are able to record data from sport activities using multiple different type of sensors, are using manually selectable sport profiles when starting a new activity recording in order to label and classify them. Because many of the sports differ significantly, it is not reasonable to record the same data for all of them. Also, in each sport different types of data are tracked during training for which case specified sport watch data fields have been created. However, there are many problems which possibly cause false sport activity labelling. The most common situations might be that 1) humans can select wrong sport profile accidentally, 2) smartwatch or recording device may not have actual sport profile, or 3) the wrong sport profile is chosen intentionally or due to indifference. This wrong labelling or unreliable human made classifying is problematic since sport activity tracking platforms have also become the one form of social media where people can discuss and compare their sport activities among each other. But also, wrong labelled activities might cause distortion in general personal data statistics, and therefore misleading guidance from smartwatch to user since modern smartwatches are highly interactive. Smart watches are continuously learning from the user, and for instance, can create training programs based on the collected data from the athlete.

1.2 Literature overview

HAR concept

The research field regarding and associated with introduced problem is called Human Activity Recognition (HAR). During the past two decades it has been widely investigated study field due to fast increasement in popularity and development of activity bracelets, smartwatches and smartphones providing sensors to measure appropriate data through inertial sensors. Advance of deep learning and machine learning algorithms has allowed researchers to use HAR in various domains including sports, health, and well-being applications. HAR is considered as one of the most promising assistive technology tools to support elderly's daily life by monitoring their cognitive and physical function through daily activities [1]. The goal of HAR is to recognize human activities in controlled and uncontrolled settings.

Sport activity classification concept

With the rapid development of wearable intelligent and artificial intelligence technologies, performance analysis in sport science has undergone major changes in recent years. Hsu, Chang, and Chiu (2019) stated that in general, manual recording and analysis performed by trained analysts in sport science has some disadvantages such as time intensive, time consuming, subjective in nature, and prone to human error and bias, and furthermore, objective measurement and analysis for sport activities is essential to understanding the technical and physical demands associated with sports performance. Sport Activity Recognition (SAR) systems are developed to provide objective measurement and analysis in sport science, which have the potential to improve the accuracy and efficiency of sports performance analysis and evaluate the effectiveness of training programs designed by coaches. Common SAR systems can be achieved through machine and deep learning approaches by using the data measured by inertial sensing technologies, where the sensors are wearable and composed of accelerometers, gyroscopes, and magnetometers. Wearable devices embedded with inertial sensors are widely used in numerous applications such as authentication, rehabilitation, gait analysis, health care, activity recognition, disease monitoring, navigation. [2]

Mekruksavanich and Jitpattanakul (2020) introduced numerous reasons for the rapid development of wearable sensor technology, namely the decreasing cost of sensor devices and the significant improvement of miniaturized sensors' computational capacity. Wearable sensors are defined as tiny devices that people are able to continuously carry during the performance of their everyday activities. Sensors such as accelerometers, barometers, GPS, gyroscopes, and magnetometers capture the signal of the physical movement of a person at any time and at any place. Further, they found that advantages of wearable sensors have been adopted in a number of beneficial mobile applications, including abnormal driving detection, healthcare systems, sport performance tracking, and mobile assistance systems for visually impaired people. [3]

Sensors and data acquisition methods

Lara and Labrador identified a generic data acquisition architecture for HAR systems, as shown in figure 1.2A. First, wearable sensors are mounted to the person's body to measure attributes of interest such as motion, location, temperature, ECG, and so on. These sensors communicate with an integration device, that is basically cellphone, PDA, laptop, or a customized embedded system. The function of the integration device is to preprocess the data received from the sensors and, in some

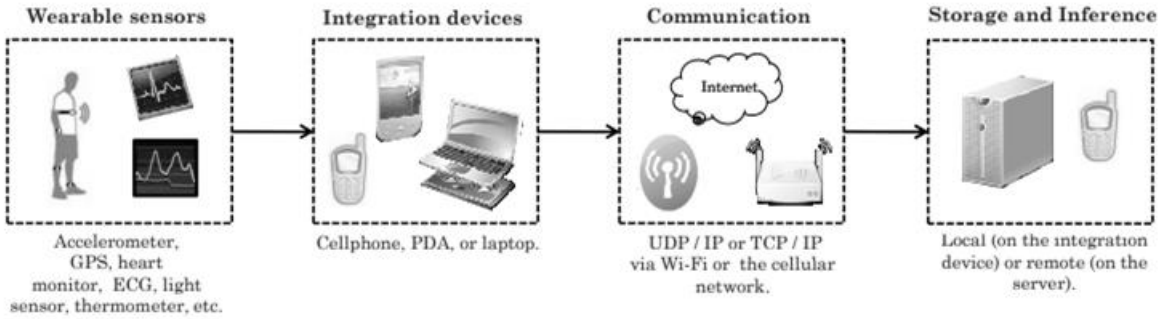


Fig. 1.2A: Generic data acquisition architecture for Human Activity Recognition.[4]

cases, send them to an application server for real time monitoring, visualization, or analysis. Nevertheless, they noted that all of these components are not necessarily implemented in every HAR system. [4]

Mekruksavanich and Jitpattanakul distinguished seven main issues pertaining to human activity recognition, namely, 1) selection of attributes and sensors, 2) obtrusiveness, 3) data collection protocol, 4) recognition performance, 5) energy consumption, 6) processing, and 7) flexibility. Then, they analyzed the main aspects and solutions related to each one of them. Furthermore, they identified four groups of measured attributes using wearable sensors in a HAR context, namely 1) environmental attributes, such as temperature, humidity, audio level, etc., which are intended to provide context information describing the individual's surroundings, 2) acceleration attributes from tri-axial accelerometers which are perhaps the most broadly used sensors to recognize ambulation activities such as walking, running, lying, etc. 3) location, the GPS that enables wide variety of location based services, 4) physiological signals, such as heart rate, respiration rate, etc. have also been considered in a few works. [3] As an example, Tapia et al. (2007) implemented an activity recognition system that combines data from five tri-axial accelerometers and a heart rate monitor. [5]

Lara and Labrador discussed HAR system obtrusiveness and stated that it should not be required for user to wear many sensors or interact too often with the application. They noticed systems which require the user to wear four or more accelerometers or carry a heavy rucksack with recording device and considered these configurations uncomfortable, invasive, expensive, and hence not suitable for activity recognition. For instance, a sensing platform that can be worn as a sport watch; a strap that is placed on the chest and a cellular phone systems are able to work with rather unobtrusive hardware. Minimizing the number of sensors required to recognize activities is beneficial not only for comfort, but also to reduce complexity and energy consumption as less amount of data would be

processed. They concluded that in many relevant works only two accelerometer setups like wrist and thigh or wrist and hip are sufficient to recognize ambulation and other daily activities. [4]

About data collection protocol, Lara and Labrador stated that procedure followed by the individuals while collecting data is critical, and the number of participants and their physical characteristics are crucial factors in HAR studies. Based on their literature overview, they concluded that a comprehensive study should consider a large number of individuals with diverse characteristics in terms of gender, age, height, weight, and health conditions, since this will ensure flexibility to support new users without collecting additional training data. [4] Furthermore, Foerster et al. (1999) demonstrated accuracy drop for ambulation activities from 95.6% of a controlled data collection experiment to 66% of uncontrolled non-laboratory natural environment. [6]

Based on the literature review, the performance of a HAR system depends on several factors, such as the activity set, the quality of the training data, the feature extraction method, and the learning algorithm. Each set of activities brings a totally different pattern recognition problem. For example, discriminating among sports such as walking, running, and biking turns out to be relatively easier than incorporating more complex human daily activities. Another aspect to consider is a sufficient amount of training data, which should be similar to the expected testing data. In addition, comparative evaluation of several learning methods is desirable as each dataset has distinct characteristics that can be either beneficial or detrimental for a particular method. Such interrelationship among datasets and learning methods can be very hard to analyze theoretically, which emphasizes the need of an experimental study. In order to quantitatively understand the recognition performance, some standard metrics are used, such as accuracy, recall, precision, F-measure, and ROC curves. Yet another important point of discussion is where the recognition task will be performed. Server is expected to have huge processing, storage, and energy capabilities, allowing to incorporate more complex methods and models, whereas a HAR system running on integration device such as smart phone should substantially reduce energy expenditures, as raw data would not have to be continuously sent to a server for processing. [4]

Because Lara and Labrador implemented comprehensive and unquestionably high-quality overview in HAR studies it is honor to point out yet some great aspects from their study regarding flexibility of model design. They identified an open debate on the design of any activity recognition model. Since according to some authors, people perform activities in a different manner as they differ on age, gender, weight, and so on, a specific recognition model should be built for each individual. This implies that the system should be retrained for each new user [7]. However, many studies rather

emphasize the need of a monolithic recognition model, flexible enough to work with different users [8]. Consequently, two types of analyses have been proposed to evaluate activity recognition systems: subject-dependent and subject-independent evaluations [5]. In the first one, a classifier is trained and tested for each individual with his/her own data and the average accuracy for all subjects is computed. In the second one, only one classifier is built for all individuals using cross validation or leave-one-individual-out analysis. Lara and Labrador highlighted that, in some cases, it would not be convenient to train the system for each new user, especially when there are too many activities; some activities are not desirable for the subject to carry out; or the subject would not cooperate with the data collection process. Older people would surely walk quite differently than a young child, thereby challenging a single model to recognize activities regardless of the subject's characteristics. They suggested that a solution to the dichotomy of the monolithic and particular recognition model can be addressed by creating groups of users with similar characteristics. [4]

Feature extraction

Machine learning requires a very large amount of data. Especially, when the dimension of the data increases significantly the data required for an accurate analysis increases dramatically. The curse of dimensionality is an important aspect to consider because sensor-based systems provide a lot of data. Therefore, it is important to select certain patterns or features, and perform classification based on these features. As the number of data increases, the computational cost also increases exponentially. That is why feature subset selection and feature extraction are used. Feature is a statistical function that works brilliantly to extract meaningful information of data in a natural way. From the perspective of human activity recognition, a particular pattern is generated from a particular physical movement of users. As an example, running activity has a particular pattern as it involves superior physical effort from a human, which is quite different from the activity pattern of walking. Thus, inertial sensors like an accelerometer and gyroscope can measure the intensity of each physical effort and produce different pattern distributions. Collected pattern distribution i.e., statistical data information from these sensors can distinguish between walking and running activity. Hence, the standard deviation or any other statistical feature is capable of highlighting the difference between these two activities. [9]

In a survey of Lara and Labrador (2013) were demonstrated main concepts about pre-requisitions in HAR data collection and training and testing systems. They stated that the training stage initially requires a time series dataset of measured attributes from individuals performing each activity. Relevant information in the raw signals is filtered by splitting the time series into time windows to apply feature extraction. Thereafter, learning methods are used to generate an activity recognition

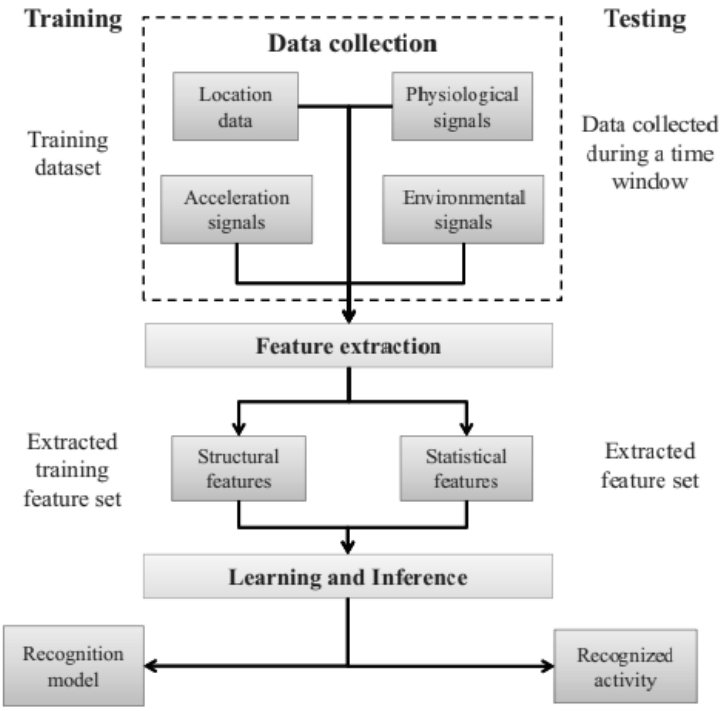


Fig. 1.2B: General data flow for training and testing HAR systems based on wearable sensors.[4]

model from the dataset of extracted features. Then similarly as for testing, data are collected during a time window, which is used to extract feature set which is evaluated in the formerly trained learning model, generating a predicted activity label. In figure 1.2 the common phases involved in these two processes are depicted. [4]

Drawbacks and challenges

According to Demrozi et al., HAR algorithms still face many challenges, including 1) complexity and variety of daily activities, 2) intra-subject and inter-subject variability for the same activity, 3) the trade-off between performance and privacy, 4) computational efficiency in embedded and portable devices, and 5) difficulty of data annotation. [1]

Micucci et al. (2017) created a dataset of activities of daily living in order to tackle lack of suitable publicly available dataset problem. They stated that data acquired by the hosted sensors are usually processed by machine-learning-based algorithms to classify human activities, and the success of those algorithms mostly depends on the availability of labeled training data that, if made publicly available, would allow researchers to make objective comparisons between techniques. Nowadays, there are only a few publicly available data sets, which often contain samples from subjects with too

similar characteristics, and lack specific information so that it is not possible to select subsets of samples according to specific criteria. [10]

Nowadays accelerometers and gyroscopes that can be employed for the classification of human activities are installed on smartphones and smartwatches. However, the focus of these efforts has mostly been on smartphone-based activity recognition. The application of these common commercially available devices significantly increased the possible uses for activity recognition; but several limits must be confessed since these devices are placed on the body of a user and the unstable orientation, such as when the smartphone moves around in a user's pocket, or when the position of the pocket is not suitable for the tracking of hand-based activities. Also, Mekruksavanich and Jitpattanakul stated that since women do not usually carry their smartphone in their pocket, they are especially prone to the limitations of smartphone-based activity recognition. Therefore, the use of smartwatches, which are worn in a constant location, addresses the majority of these drawbacks, as they are ideally positioned for the tracking of activities that are hand-based. Nevertheless, according to them, these approaches using conventional machine learning regularly depend upon heuristic manual feature extraction and are thus normally limited to the knowledge of the human domain. They stated that these limitations restrict the performance of approaches using conventional machine learning in terms of the accuracy of the classification. [3]

Methods, applications, and results in HAR

In terms of algorithmic implementation, according to Demrozi et al. (2020) HAR research has seen an explosion in Deep Learning (DL) methods, resulting in an increase in recognition accuracy. While DL methods produce high accuracy results on large activity datasets, in many HAR applications Classic Machine Learning (CML) models might be better suited due to the small size of the dataset, lower dimensionality of the input data, and availability of expert knowledge in formulating the problem. The increasing interest in HAR can be associated with growing use of sensors and wearable devices in all aspects of daily life, especially with respect to health and well-being applications. [1] Data for training and testing HAR algorithms is typically obtained from two main sources such as *ambient sensors*, and *embedded sensors*. Ambient sensors can be environmental sensors such as temperature sensors or video cameras positioned in specific points in the environment. Embedded sensors are integrated into personal devices such as smartphones and smartwatches, or correspondingly integrated into clothes. [2]

On literature overview of Hsu, Chang, and Chiu (2019) for HAR studies, they found that a number of researchers have developed wearable inertial-sensing-based sport activity monitoring and

analysis systems for running, football, tennis, baseball, golf, fencing, basketball, cycling, badminton, table tennis, volleyball, and many others. They identified several signal processing procedures required to generate a suitable input data for machine learning classifier algorithms to recognize sport activities, which include filtering, window motion durations, signal normalization, feature extraction, feature reduction/selection, and classification. For example, one may select the 11 acceleration features from the 13 most commonly time and frequency-domain features by using the Relief method, which are input to the Naïve Bayes (NB), Decision Tree (DT), Logistic Regression (LR), and Artificial Neural Network (ANN) classifiers for recognizing the 8 sport activities including cycling, cross trainer, rowing, running, squatting, stepping, walking, and weightlifting. Another developed method utilized the custom decision tree combined with the ANN to classify the 9 sport activities which consist of lying down, sitting and standing, running, walking, rowing with a rowing machine, cycling with an exercise bike, nordic walking, playing football, and cycling with a regular bike, using the 7 time- and frequency-domain features extracted from the accelerometers and GPS measurements. In the literature review of Hsu, Chang, and Chiu they stated that many researchers in the field of machine learning have paid attention to the inertial-sensing based sport activity classification tasks. However, there is a few literatures that utilizes deep convolutional neural networks (CNN), which can extract inherent features from the inertial sensing measurements, to recognize sport activities. Hsu, Chang, and Chiu implemented a wearable sport activity classification system and its associated deep learning-based sport activity classification algorithm for sport activity classification task. Their developed method for wearable sport activity classification system consisted of two wearable inertial sensing modules worn on athletes' wrist and ankle to collect motion signals of sport activities. The proposed deep learning-based sport activity classification algorithm composed of sport motion signal collection, signal preprocessing, sport motion segmentation, signal normalization, spectrogram generation, image mergence/resizing, and CNN-based classification was developed for classifying ten types of sport activities divided into table tennis, tennis, badminton, golf, batting baseball, shooting basketball, volleyball, dribbling basketball, running, and bicycling. The goal was to develop a low cost wearable sport activity classification system and its deep learning-based sport activity classification algorithm for providing an effective tool for sport activity classification tasks, which utilizes the deep CNN classifier for obtaining more accuracy for sport activity classification. Their experimental results show that the proposed wearable sport activity classification system and its deep learning-based sport activity classification algorithm can recognize 10 sport activities with the classification rate of 99.30%. [2]

On the study of Mitchell et al. (2013) was presented the discrete wavelet transform (DWT)-based support vector machines (SVM) optimized by the sequential minimal optimization (SMO) algorithm to recognize the 7 sport activities using the measurements of the smartphone accelerometers. [11]

Wang et al. (2018) used the principle component analysis (PCA) to obtain 3 features from the 12 statistical features and 3 morphological features extracted from the microelectromechanical systems (MEMS) motion sensors' signals, which are input to the SVM classifier for classifying elite, subelite, and amateur volleyball players with an average accuracy of 94 %. [12]

Hsu et. al. (2018) developed the activity recognition algorithm, consisting of procedures of motion signal acquisition, signal preprocessing, dynamic human motion detection, signal normalization, feature extraction, feature normalization, feature reduction, and activity recognition, to recognize human daily and sport activities by using accelerations and angular velocities. In order to reduce the computational complexity and improve the recognition rate simultaneously, they utilized the nonparametric weighted feature extraction algorithm with the principal component analysis method for reducing the feature dimensions of inertial signals. In their study, all 23 participants wore the wearable sensor network on their wrist and ankle to execute 10 common domestic activities in human daily lives and 11 sport activities in a laboratory environment, and their activity recordings were collected to validate the effectiveness of the proposed wearable inertial sensor network and activity recognition algorithm. Their experimental results showed that proposed approach could achieve recognition rates for the 10 common domestic activities of 98.23% and 11 sport activities of 99.55% by the 10-fold cross-validation strategy, which have successfully validated the effectiveness of the proposed wearable inertial sensor network and its activity recognition algorithm. [13]

Hsu et al. (2018) conducted a comprehensive survey on recent HAR studies about implemented methods on the basis of their study in daily and sport activity recognition using a wearable sensor network. They found that many researchers have focused on developing effective human daily activity recognition systems through combining various feature reduction methods with machine learning classifiers to measurements collected from wearable devices. They named several methods developed in previous studies such as 1) Kernel Discriminant Analysis (KDA) that is based on Extreme Learning Machine (ELM) for recognizing daily human physical activities using an accelerometer placed on the subject's thigh. 2) Principal Component Analysis (PCA) method to reduce the input features from 1170 to 30 to represent a motion signal sample, which were captured from the human activity signals

measured by the MTx trackers mounted on the user's wrist, knee, and chest in a laboratory environment. Subsequently, the 30 PCA-based features were treated as the input features for the Bayesian decision making classifier. 3) Common PCA (CPCA) combined with the support vector clustering (SVC) algorithm to develop the feature subset selection (FSS) algorithm for the 24 features extracted from the static and dynamic activities, which were collected in a laboratory environment. The 8 common domestic activities composed of standing, sitting, walking, running, vacuuming, scrubbing, brushing teeth, and working at a computer were recognized by the FSS combined with the feedforward neural network (FNN). 4) The KDA-based support vector machines (SVMs) were utilized to recognize 15 human activities in a laboratory environment using the measurements of the accelerometer, pressure sensor, and microphone. 5) Dynamic Linear Discriminant Analysis (LDA) combined with the fuzzy basis function (FBF) classifiers for classifying 8 human daily activities in a laboratory environment with satisfactory recognition accuracy. 6) Hybrid classifier was utilized combining the tree structure and the artificial neural network (ANN) to recognize 9 daily and sport activities. The accuracy for categorizing 9 daily and sport activities using both supervised and unsupervised data was 89% by the 12-fold leave-one-subject out cross-validation. 7) Recognition performance of the template-matching and statistical-learning classifiers was compared for recognizing 8 common sport activities using a tri-axial accelerometer wore on users' wrist. [13]

Further, Xie et al. (2018) proposed a HAR method based on inertial sensors and barometer. Their developed method recognized eight human activities following a multi-layer strategy, wherein activities were classified into dynamic and static activities; and then explicit activity recognition was taken individually in the two categories. Three classifiers were adopted for different classification, including random forest (RF) and support vector machine (SVM). They selected different feature sets for different classifiers which were more targeted and effective. Classifier result was verified by additional parameters and previous recognition results to decide the final recognition result. [14]

In the study of Ghazali et al. (2018) common sport activities such as stationary, walking, jogging, sprinting, and jumping were classified using inertial sensor data. They extracted time-domain features such as mean, standard deviation, maximum and minimum values from the signal produced by inertial sensor to recognize common sport activities. After that, the data was trained using the selected features and 10-fold cross validation was applied to set the training and testing data. The confusion matrix and accuracy result from each classifier was observed They concluded cubic SVM as the best model in recognizing the common sport activity as it produced highest accuracy which was 91.2%. [15]

Fang et al. (2016) extracted features from raw data and processed them using support vector machine (SVM), K-nearest neighbor, and logistic regression. Average accuracy for SVM was 88.9%, KNN was 95.3%, and Logistic Regression was 83.9%. The paper mainly focused on up and down buses. For these two activities, accuracy was 89.5% for SVM, 96.8% for KNN, and 89.7% for LR for up bus, and for down bus, 84.2% for SVM, 94.2% for KNN, and 81.9% for LR. [16]

Yin et al. (2015) utilized four machine learning methods which were J48, SVM, naive Bayes, and Multilayer Perceptron (MLP) for detecting five activities. The processing step was divided into four subject areas namely smartphone-based approach; data collection; feature extraction; and classification. Data were collected in a three-axial accelerometer, three-axial linear accelerometer, gyroscope, and orientation. J48 is an algorithm, in which the output is produced in a decision tree and accuracy was 96.8%. Accuracy of others based on five activities was 98.7%, 97.8%, and 98.5%. The accuracy of all methods was over 99% when using only three basic activities. They concluded that J48 is more efficient due to easy computing and because it can be converted into if-then clauses. [17]

The research of Ahmed et al. (2020) presents hybrid feature selection model-based smartwatch sensor data, which robustly identified different human activities. The data were gathered from inertial sensors, accelerometer and gyroscope and mounted on the waist of the objects, like humans. The research was implemented by recording different human activities by the researchers in a research lab. They applied a total of 23 base features were applied on the dataset and extracted 138 heterogeneous features from the sensors. They found that not every feature contributes in the same way for activity recognition, but extraneous features degrade the performance of the classifier. Therefore, their proposed hybrid feature selection method containing the filter and wrapper approaches played an important role for selecting optimal features. The selected features were used for validation test using the SVM to identify the human activities. The proposed system conducted 96.81% average classification performance using those selected optimal features, which produced around 6% improvement in performance compared to no systems without feature selection. [9]

Rassem et al. (2017) studied performance of deep learning methods in cross-country skiing gears classification and addressed that they are more effective with highest classification accuracy. They stated that the MLP failed to have a high performance in the classification due to the complex nature of the input skiing signals which have varying intensities and frequencies because skiers have different styles to perform the same gear. Both stacked layers and the non-linear complex functions used by deep models enable CNN and LSTM to achieve good results represented in low classification error. In their study CNN had the minimum error of 2.4% using a small number of local filters for the

convolution process where increasing the filters does not necessarily improve the performance. The standard forward LSTM had the lowest classification error with 1.6% score over all the five models due to its recurrent architecture which utilize shared weights through all time steps. [18]

1.3 Thesis introduction

Concepts

In this thesis I will regularly use a new concept Sport Activity Classification (SAC), which refers to the same concept as previously mentioned SAR, and its expansion Time Series Sport Activity Classification (TS-SAC), which will be sub-concepts of HAR. In general, they refer to the same base idea of interpreting inertial sensor data derivatives over time (Time Series) or signal statistics. In HAR tasks, usually ideal and in laboratory environment carefully measured data is used in order to train a model for activity classification in test data. However, in practice such preconditions are often hardly available and raw data for analysis will be something totally different. Dataset will be collected by repeating some activities (series of movements) X times, done by different persons. Basically, that means dataset does not contain, for example, mixed movements where we have to select between two or more activity types, and that makes classification a very straightforward process. For inertial sensor measurements this method has succeeded quite well producing tolerable results in several previous studies in last decade, and discriminatory patterns have been found among different type of human activities [1], [19]. In their comprehensive survey on HAR studies Demrozi et al. summarized that recent studies have yielded quite good results with an average accuracy of 93% using deep learning methods and almost as good results with classical machine learning methods 92.2% among a total of 149 papers on HAR, published from 2015 to 2019. They stated that while DL methods produce high accuracy results on large activity datasets, in many HAR applications CML models might be better suited due to the small size of the dataset or lower dimensionality of the input data. Further, they noticed that the main problem in the development of HAR algorithms has been the relatively difficult public availability or absence of high-quality datasets in the subject. The same was stated in the study of Micucci et al. (2017) where they claimed that the success of machine learning algorithms mostly depends on the availability of correctly labelled training data that would allow researchers to make objective comparisons between methods if data is publicly available. According to them, there are only a few publicly available data sets, which often contain samples from subjects with too similar

characteristics, and very often lack specific information. While conducting this thesis the same problematic premise of lacking versatile data for endurance sport activity recognition is acknowledged. [10]

This study consists of comprehensive CML model performance evaluation for sport activities. We use three different type of datasets which vary in structure and features. The first type of data for traditional CML is summary data of the activity such as distance, maximum, average, and minimum values for wide variety of features from the sensors. The second type of data is univariate time series sequences where also instance order contains significant information, and in which we apply certain multivariate data transformation technique to univariate in order to include more information to the signals. The third type of data is multivariate time series sequences in which we apply specific models and column ensemble classification. Thus, study is divided into three sub-analyses according to different type of datasets and therefore CML models to be applied. These are Standard Classical Machine Learning classification (S-CML), Univariate Time Series Classification (U-TSC), and Multivariate Time Series Classification (M-TSC).

Standard CML

For S-CML we use the dataset that contains 21 features and most of them are extracted features from the same sensor. Hence, derived features are used instead of raw data and in that sense this study differs from traditional ones which basically are performed in three-axial, three-dimensional sensor data as introduced earlier in the thesis. The dataset for S-CML consists of 220 activities of single active athlete covering a total of five different sport types. Emphasis will be set on careful raw data pre-processing which consists of data formatting, manual feature selection, standardization, and transformation to achieve preconditions for successful S-CML model analysis. In classification, we use 70-30% division in training and test data splits due to the small size of the dataset. The most common S-CML algorithms which have performed well in previous corresponding sport or HAR studies are used in this thesis. According to the recent comprehensive survey of Demrozi and others, k-Nearest Neighbor (kNN), Support Vector Machine (SVM) and Decision Tree (DT) has been the three most general well performed CML models used in activity recognition tasks. Other applied S-CML methods are Random Forest (RF), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Logistic Regression Classifier (LR), and Gaussian Naïve Bayesian (G-NB). Also, Multilayer Perceptron (MLP) classifier is used as representative of neural network technologies. Most of the used methods have performed with similar accuracy in referenced studies. [19], [20], [21].

Time Series Classification

Time Series Classification (TSC) is a subfield of machine learning with numerous real-life applications. Due to the temporal structure of the input data, standard CML algorithms are usually not well suited to work on raw time series. Over the last decades, many algorithms have been proposed to improve the predictive performance and the scalability of state-of-the-art models. [22]. Potentially the most suitable ones of those models will be applied in TS-SAC. Data from the Heart Rate sensor (HR), GPS sensor (Speed), and Barometer sensor (Altitude) are used in both univariate and multivariate TSC. The most important and challenging part of the U-TSC is to prepare and construct multivariate data in such manner that it allows to conduct classification in univariate space. In traditional SAC, dataset is structured so that rows consist of summary details of the activities, that means usually maximum, minimum, average, etc., values for the features. Such a dataset is relatively small and does not set unreachable requirements for computational resources. Whereas in TS-SAC, task is expanded to use the whole time series data from the sensors for each single activity, in principle. The natural consequence is that processed data amount will be increased significantly compared to traditional SAC. Although only three features are used in TS-SAC which will reduce the amount of the data to a fraction, the same time complexity remains.

Univariate TSC analysis consists of eight univariate *sktime* classifiers, Time Series Forest Classifier (TSF), Supervised Time Series Forest (STSF), Random Interval Spectral Forest (RISE), Random Interval Classifier (RIC), Shapelet Transform Classifier (ST), k-Nearest Neighbours Time Series Classifier (kNN-TS), Composable Time Series Forest Classifier (CTSF) which is like TSF but has extended composability, and Word Extraction for Time Series Classification (WEASEL). Also, we conduct Hierarchical Vote Collective of Transformation based Ensembles (HIVE-COTE) as an ensemble for univariate data which is a method to combine several base classifiers using advanced ensembling technology by weighting class probabilities of models.

In M-TSC study, we conduct Multivariate Symbolic Extension (MUSE) method and build an ensemble model to classify each attribute distinctively, using optimal model and then combining predictions to conduct final classification. The expectation is that we achieve the best results with these methods, because the transformation of multivariate data into univariate data is often practically difficult to implement without loss of information. Although dimensionality reduction can simplify the classification task, and therefore, sometimes make it more effective in terms of time.

In addition to TSC models, we feed the same data constructed for U-TSC to our selected S-CML models, expanded by Gradient Boosting (GB), in tabularized format for two reasons: at first, to

create a reference point and a reflection surface for the TSC results, and second, to assess the functionality of the interlaced three dimensional data structure at a more general level by classifying both dataset types of this thesis using the same models.

Model validation and objectives

In S-CML the selected models are validated by using 5-Fold Cross validation (A-3.1B). Considering the premise, the main objective is to find at least one ML model that achieves an error-free result in the particular selected data set which contains relatively little room for theoretical interpretations between different sports, and therefore either represent the average practical reality.

Lastly, it is reasonable to mention a few important facts to pay attention to in this thesis. Dataset consists of sport records from the single athlete and therefore this study does not take into account interpersonal variations. For example, Rokni et al. (2018) implemented a neural network architecture using transfer learning techniques which enables them to build a personalized HAR model with minimal human supervision. [20] This minimal human supervision is the ultimate goal of personalized HAR algorithms and in that sense this study is quite primitive and only framework for future studies. Further, the motivation to investigate SAC-problem from this point of view is mainly targeted to fitness and active athletes and other sport enthusiasts which will record their activities using any device and upload data into the cloud service to be compared among others, for example, in segment leader boards.

DATASET CONSTRUCTION

2.1 Dataset for Standard CML

Dataset origin and devices

Dataset for S-CML consists of sport recordings from a single athlete covering 16 month of time period. Activities are recorded by two Garmin sport watches FR-920XT and Vivosport (A-DEVS), latter also called activity tracker. Model FR-920XT has integrated GPS and barometer sensors but external Heart Rate (HR) and inertial sensor mounted right below the chest. Whereas Vivosport uses integrated wrist based heart rate monitoring, having more instability. This HR-Inertial sensor enables to measure running dynamics including vertical oscillation, vertical ratio, cadence, and many other derived features, but because function is enabled only for running activity, these features cannot be applied in classification task. Records from the devices are uploaded to the Garmin Connect (GC)

cloud service after finishing the activity and the dataset is constructed from that raw data. Since other studies do not pay attention to dataset construction process but rather try to dodge and tackle problems by creating flexible algorithms, in this thesis careful data formatting, and further, preprocessing is emphasized in order to get best possible premise for successful sport classification. We start by taking a quick overlook for the raw data and features and then proceed by reporting main issues encountered during data formatting and finish in final feature exclusion.

Raw data formatting and feature exclusion

Original dataset A-2.1A from GC cloud consists of 220 rows and 40 columns and is saved in *csv* format. The first step is to remove columns which are not relevant and can be clearly seen that they do not provide any relevant information related to activity classification task. After this, there are 27 features left in the dataset before single cell value formatting in precondition to a column value type transformation from *string* (object) value to a numeric. In time data formatting, where colons or dots separated (h:m:s/h.m.s) values will be transformed to a floating point number. Then we need to replace all commas by dot in the dataset and remove thousand separators from features. Due to the small data dimensionality, columns can be detected manually. This will be implemented in the whole dataset by using algorithm which selects all the object value type columns from the dataset. For the *calories* feature, integer type conversion was also made. There are also two time-type data columns in the unit of pace, wherein different activity types are using different units. In this case, pace data can be expressed as a speed value. We solve this before moving all the rest of the object columns to a numeric format. In supervised classification it is crucial to have enough instances for every activity type for both train and test splits. Therefore, class labels less than intuitively selected criteria (20) are rejected. All the instances for four different activities were dropped out, and also, all related features according to single unique value rule. Thus, after raw data formatting and knowledge based feature exclusion dimensionality of the dataset has decreased to 22 columns, including label column used in classification task.

Dataset preview

SAC is performed in the dataset described in the appendix A-2.1B. There are 22 features and 213 instances which will be considered in a principal component analysis when selecting the most effective features. Target column is *Activity Type* feature, which is the only categorical value column. Also, there are two integer value columns, but all the rest are a type of floating value. Correlation

matrix shows descriptive overview of the nature of the data features and their relations (A-2.1C). Lastly, the most significant factor is the target feature class division between activity types that can be seen on the figure 2.1A. There are 5 types of activities to be recognized in classification task. Pair plot A-2.1D provides quite clear representation about how five first features of the dataset behave in relation to each other among activity types.

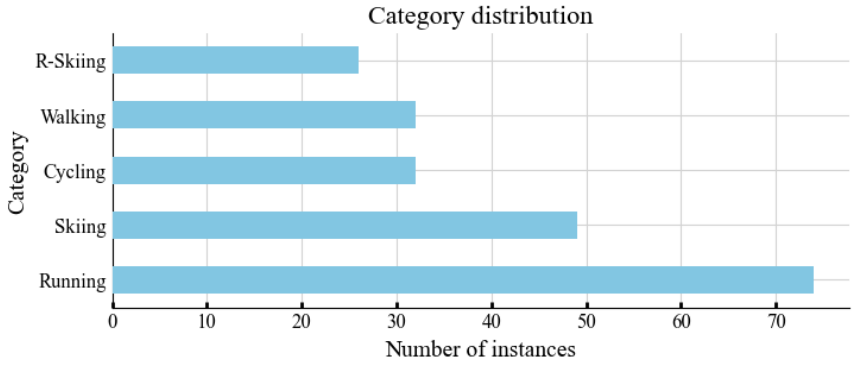


Fig. 2.1A: Category distribution among activities.

Data transformation, PCA and t-SNE

After rough data processing the dataset is prepared for further investigations and data transformation. Since all the columns are non-categorical, floating or integer value type columns, standardization process will be successful without data distortion. Dataset contains variables or features of different scales. In order for our machine learning or deep learning model to work well, it is very necessary for the data to have the same scale in terms of the feature to avoid bias in the outcome. Thus, feature scaling is considered an important step prior to modelling. Standardization technique follows the formula $z = \frac{x-\mu}{\sigma}$,which will set the mean value of the dataset features to zero (0) and to have unit variance. Standard deviation equals to one (1).

Two techniques were used to investigate how well activity types of the dataset will be separated into clusters: Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE). PCA is a fundamentally a simple dimensionality reduction technique that transforms the columns of a dataset into a new set of features called Principal Components. The information contained in a column is the amount of variance it contains. The primary objective of Principal Components is to represent the information in the dataset with the minimum columns possible. Practically PCA is used for two reasons: *dimensionality reduction* and *class visualization*. A t-SNE is an unsupervised, *non-linear technique* primarily used for data exploration and visualizing high-dimensional data

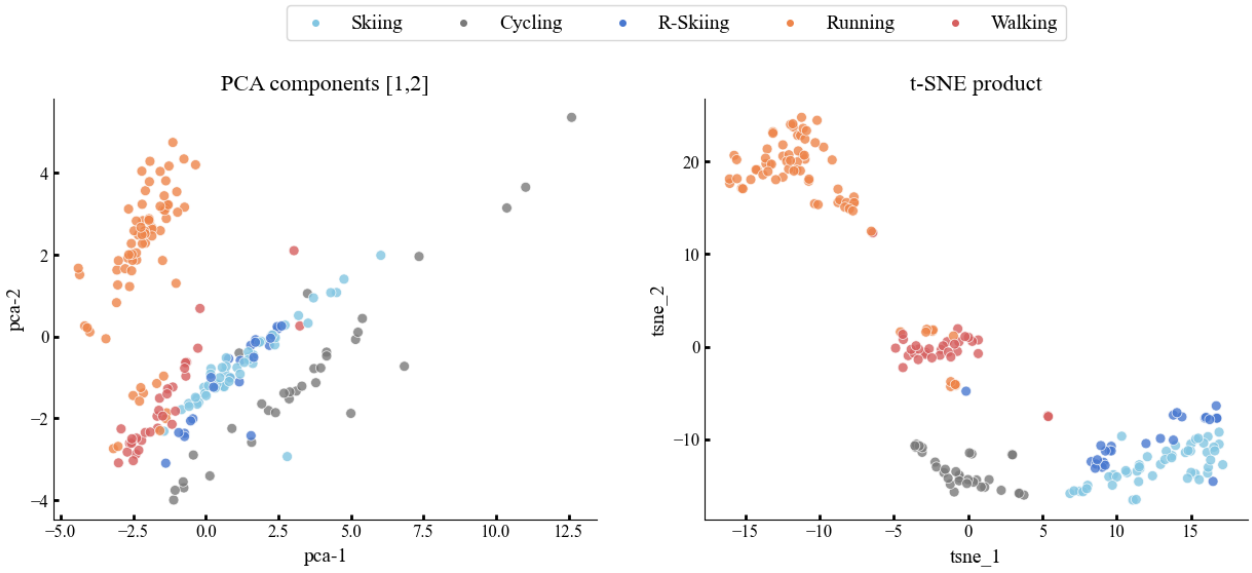


Fig. 2.1B: PCA and t-SNE data visualizations. Two potential main problems in classification can be noticed. Confusion between running and walking activities, and similar situation between Skiing and R-Skiing (Roller Skiing).

and gives an intuition of how the data is arranged in a high-dimensional space. When comparing these two techniques, the first thing to note is that PCA was developed in 1933 while t-SNE was developed in 2008. PCA focuses heavily on linear algebra while t-SNE is a probabilistic technique. Also, PCA is *a linear dimension reduction technique* that seeks to maximize variance and preserves large pairwise distances. [23] Inspecting t-SNE activity analysis results in figure 2.1B, two potential main problems in classification can be noticed. Confusion between running and walking activities, and similar situation between *Skiing* and *R-Skiing (Roller Skiing)*.

Corresponding standard data scatter plot is available in appendix A-2.1E. Classification is performed in both PCA and standardized data, but t-SNE is not considered later in this thesis. For PCA, *explained variance ratio* metric was used to select minimum set of principal components to be used in classification tasks although its advantage in this small dataset case may not be significant. Figure 2.1C depicts that six most principal components will explain ~90% of the variance in the feature space.

Pair plot visualization among 5 most principal components shows, in general, that activity types are quite well separable into clusters (A-2.1F), and we can expect satisfactory results from most of the machine learning models. Further, some of them are expected to provide perfect classification, error-free results in our particular dataset that is set as a goal for this study.

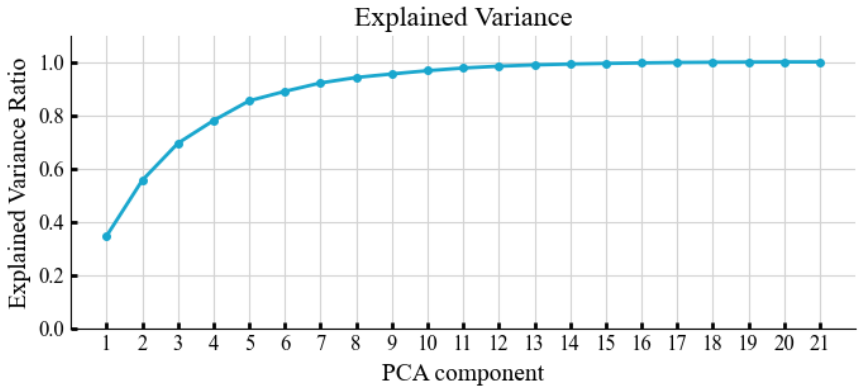


Fig. 2.1C: Explained variance ratio among principal components.

2.2 Time Series dataset

Dataset construction for TSA

Since the dataset for TSC consists of a set of separate files the question is how to construct a unified, consistent, and complete dataset for classification task. In the very first step we must proceed carefully for both, in order to ensure successful classification among activities, and also to get reliable and stable results for executions in several kind of data and TSC model setups. There exist a few methods to form a dataset from the raw files for U-TSC. The initial dataset used in this study consists of very challenging data which requires a lot of pre-processing and data cleansing. It should also be noted that the data may not be categorized correctly by the user, and in that sense does not even provide the conditions for comparability to those studies where the data has been collected under laboratory conditions. However, we know that the level of correctness is approximately at least 95%.

The dataset is created by a single athlete using Garmin Forerunner 925XT smartwatch model with external heart rate sensor mounted right below the chest as in the S-CML dataset. It consists of 297 activities for 16 months in the years 2020 and 2021. These activities have varying lengths from 0 to ~20000 seconds. There are 5 different sport categories, which are running, biking, skiing, roller skiing, and walking. However, in this case we simplify classification task a little bit and implement three class sport recognition task using *running*, *biking*, and *other* categories. That means, class *other* consists of the rest of sports except running and biking. The decision for simplifying is a technical matter related to how data is stored in Garmin Connect cloud. Further, we select only three main features, namely *heart rate*, *speed*, and *altitude* which can be considered as the most significant and relevant features for classification. Motivation is also mainly to use exactly these features and to explore how well we are able to classify sport activities from these particular features.

In the following sections we introduce data as a folder structure and preview a couple of activities. Then we go through the steps of compiling the dataset from varying length raw data files to a cleaned and processed equal length segment data with labels. In final dataset we split every single activity into $M = 69$ seconds intervals which implies *interlaced univariate sequences* in the length of 207, when using three mentioned features. Later in this thesis we use M to describe selected sequence length. The length of 69 seconds was selected by balancing computation time and accuracy. Using longer sequence may not increase the accuracy significantly compared to the required time for classification. For example, using twice as much time with only 1 percent improvement in accuracy has no significance in our classification case. In the other hand, using shorter sequence may decrease the accuracy too fast in relation to achieved computation time benefit. However, there might still be room for sequence length optimization in future studies.

Directory

Single sport activities are downloaded from the Garmin Connect cloud service in the format of Training Center XML (TCX). TCX is a data exchange format introduced in 2007 as part of Garmin's Training Center product. The XML is similar to GPX since it exchanges GPS tracks but treats a track as an Activity rather than simply a series of GPS points. TCX provides standards for transferring heart rate and other data features in the detailed track. To transform these TCX files to Comma Separated Values (CSV) files, particular algorithm was developed (A-2.2A). Also, activity labels were extracted from the TCX metadata when operating TCX to CSV conversion, and a single file containing these targets was constructed. Thus, we resulted with a set of distinct CSV files and the file with labels in which labels followed the actual file order in the folder. Distinct files in the folder are ordered chronologically.

Dataset preview

In this section we pick a single random sample activity from the dataset, in order to understand the essence of data. Each activity file consists of nine features from which we use *datetime* as an index and are going to select heart rate, speed, and altitude to construct three dimensional sequences, which we will discuss later in the next section *Data sequencing*.

In the data sample in table 2.2A, there are 6815 rows which means that activity has the length of 6815 seconds at 1 Hz recording frequency. As we are going to select only three features, namely heart rate, speed and altitude, those signals are visualized in figure 2.2A. In the x-axis we use datetime as it was set as an index value.

Table 2.2A: Single activity data sample.

Datetime	Latitude	Longitude	Altitude	Distance	HearRate	Speed
2020-05-25 15:58:03+00:00	61.432712	23.803714	128.800003	1.930000	92	1.260
2020-05-25 15:58:04+00:00	61.432697	23.803715	128.800003	3.660000	97	1.176
2020-05-25 15:58:05+00:00	61.432684	23.803711	128.800003	5.030000	99	1.138
2020-05-25 15:58:06+00:00	61.432662	23.803709	128.800003	7.410000	99	1.344
2020-05-25 15:58:07+00:00	61.432642	23.803704	129.199997	9.650000	101	1.278
…	…	…	…	…	…	…
2020-05-25 17:51:23+00:00	61.432319	23.803627	120.800003	30138.349609	169	0.000
2020-05-25 17:51:24+00:00	61.432319	23.803627	120.800003	30138.349609	168	0.000
2020-05-25 17:51:25+00:00	61.432319	23.803627	120.800003	30138.349609	166	0.000
2020-05-25 17:51:26+00:00	61.432319	23.803627	120.800003	30138.349609	165	0.000
2020-05-25 17:51:27+00:00	61.432319	23.803627	120.800003	30138.349609	162	0.000

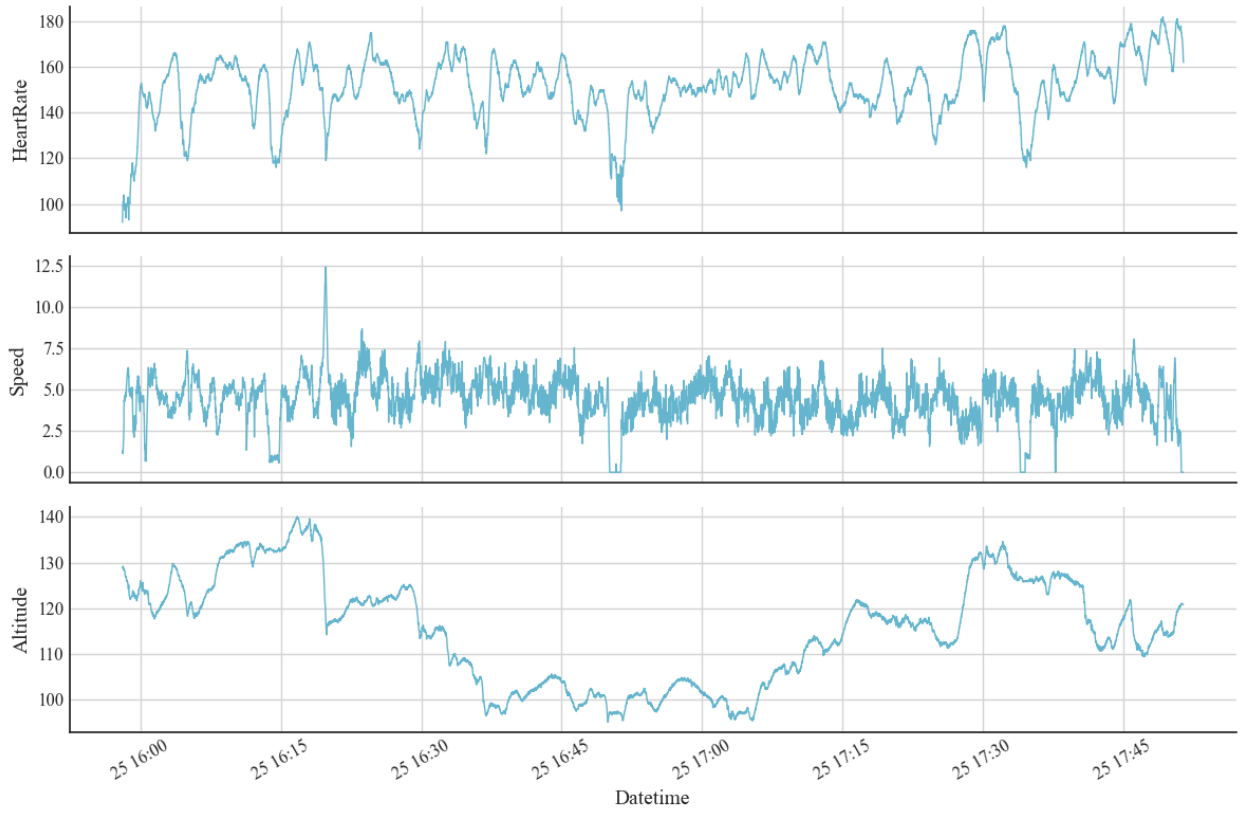


Fig. 2.2A: Heart Rate, Speed, and Altitude signals in some random activity of the dataset.

Data can be visualized quite descriptively for the context of outdoor sport activity by using *longitude* and *latitude* attributes. In the figure of data sample visualization 2.2B, data is presented in

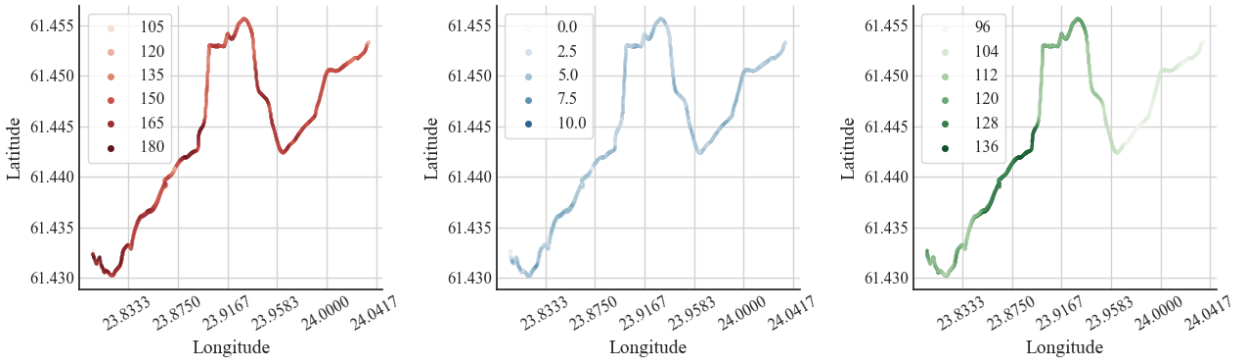


Fig. 2.2B: Data sample visualization. Figure series depicts the value intensity of three features in longitude-latitude space for the same activity. From left to right: Heart Rate, Speed, Altitude.

geolocation space by setting longitude and latitude for the axes x and y, and in z axis one of third features, for instance, one of those selected features for TSC in this study case. Darker colour indicates higher intensity of *heart rate*, *speed*, and *altitude* attributes. In the project file there are three-dimensional charts for the same sample data, that can further contribute understanding of the base behavior of the data. See the appendix A-2.2B, where it is shown for the same random activity sample as in the figure 2.2B.

Data Sequencing

First of all, we have to make clear what kind of data is to be examined and what data sequence means as a term. Traditional machine learning assumes that data points are dispersed independently and identically, however in this case we have time series data in which one data item is dependent on those that come before or after. In other words, when the points in the dataset are dependent on the other points in the dataset, the data is termed sequential. In sensor data, each point reflects an observation at a certain point in time. Data sequencing is the process of producing a sequence of values from a set of input values.

In the context of data sequencing data was cleaned using the code A-2.2C. Entry list contains all the filenames (297), and each file will be read into Pandas data frame for pre-processing. Particular desired features are selected, namely *heart rate*, *speed*, and *altitude*. A significant question of data pre-processing is how to fill gaps in the data. We instinctively selected linear interpolation method as the most descriptive, efficient, and sufficient for the context. Generally, linear interpolation takes two data points, say $(x_a, f(x_a))$ and $(x_b, f(x_b))$, and the interpolant is given by:

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) \text{ at the point } (x, f(x))$$

Since there are gaps in the very beginning, and probably in the end of the sequence, bidirectional linear interpolation was applied. After data sequencing part we have three-dimensional data in the shape of $(297, ?, 3)$ in which the question mark indicates varying length sequences. Sequence statistics are as in table 2.2B.

Data Filtering

However, this data has potential problems in classification since we have not processed empty or zero length, too short, missing sensor data, or other types of incomplete data. After applying filter algorithm with criterion parameters minimum length equals 500 and setting minimum average value requirements for each of the three features, 65 sequences were rejected. Average speed and altitude limits were set to 1, and heart rate to 100. These limits indicate basically whether corresponding sensor has been installed during activity or not. The heart rate limit set as 100 is a fairly good indicator to discriminate between inactive and active state for a particular person. Inactive state types of activities are not interesting in our study case. The results of sequence filtering and cleaning can be seen in table 2.2B. The final data shape to be used in further analysis is $(232, ?, 3)$, where the question mark indicates some varying sequence length value between minimum and maximum, in the value range of $[524, 19781]$.

Table 2.2B: Sequence/Signal length statistics before and after applying data filtering.

Feature	Original data	Clean data	Change %
count	297.000	232.000	-21.89
mean length	3952.636	4624.396	+17.00
standard deviation	3122.473	2797.697	-10.40
min length	0.000	524.000	-
25%	1223.000	2906.750	+137.67
50%	3817.000	4452.500	+16.65
75%	5643.000	5978.000	+5.94
max length	19781.000	19781.000	0

Data Segmentation

In this thesis *segmented data* refers always to the *data augmentation* technique. That means dividing single activity into several, equal length segments instead of getting only one segment per activity. By using this method, we are able to increase data instances for each category and thus probably stabilize classification results in a relatively small dataset. Segmentation of sequences was implemented with predefined parameters such as segment start, length, and number (100, *M*, 5), wherein

M was set to 69 as described previously. Start of each activity is considered to be non-reliable data based on the assumption, that recording is basically started in the very same kind of state. That usually means zero speed, for instance, and therefore a couple of minutes are needed to achieve distinctive behavior between sports. In practical terms, it is highly likely that an athlete starts recording of sport activity 1-2 minutes before one actually begins to run or cycling. When continuing with filtered data from the previous section in the shape of (232, ?, 3) we resulted with equal length sequence segments in the shape of (232*5, M, 3) = (1160, M, 3), having data from the original sequence interval $[100, 100 + 5xM] = [100, 445]$, when $M = 69$. To be clear, all the rest of the data is simply swept aside. The main reason for that is computational, but also because it does not provide significant effect to the classification accuracy. Thus, the segmentation formula can be expressed as follow:

$$D_{seg} = \bigcup_{i=0}^{N} D_i(s, l, n)$$

where $D = original\ filtered\ data, N = number\ of\ activities$, $s = segment\ start\ index$, $l = segment\ length$, and $n = the\ number\ of\ segments$ from a single original sequence D_i, resulting the new set of equal length segments D_{seg}. And in this certain case the equation takes the form of:

$$D_{seg} = \bigcup_{i=0}^{232} D_i(100,69,5)$$

and therefore D_{seg} the shape of $D_{1160x207}$.

Dataset construction and standardization

The most important and challenging part of the TSA classification is to prepare and construct data in such a manner that it can be processed. Therefore, the most significant question is how to construct dataset from raw data into readable format for *sktime* TSC models. Until this point, we have proceeded quite straightforwardly in data pre-processing, but the next step is probably the most crucial. There are several ways to construct consistent dataset for time series classification models. In the solution applied we use nested data structure for TSC, as proposed in the study of Löning et al. (2019), wherein three-dimensional signals are first divided into M seconds sequences and then flattened using interlacing method. Then those flattened sequences were combined into the same two-dimensional data table in the shape of $(1160, 3xM) = (1160, 207)$, when $M = 69$. Thus, in fact, each row value now represents feature value in the reshaped data structure, repeating the pattern *heartrate-speed-altitude* M times. Sktime classifiers are expecting data in nested data structure whereas classical

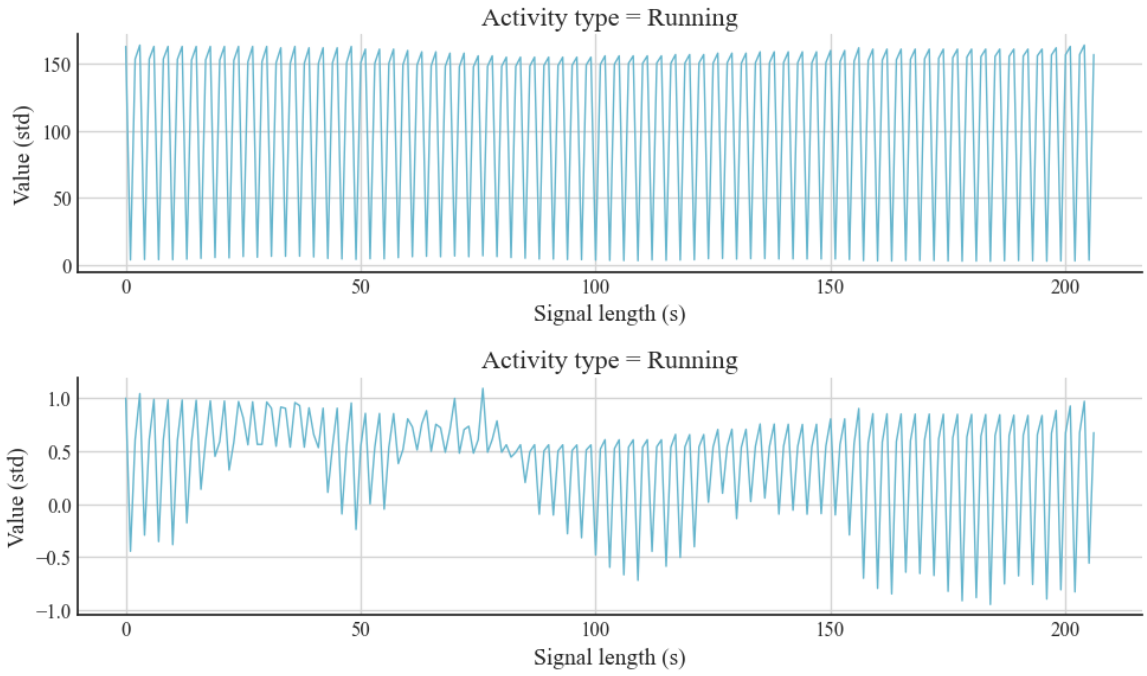


Fig. 2.2C: Interlaced signal standardization. Above non-standard interlaced univariate signal visualization for a sequence sample. Below the same sample is standardized.

machine learning algorithms require tabularized data. In nested data structure, all the features of the flattened sequences will be compressed into one dimension. In this kind of structure, one table cell consists of 207 values, according to second dimension length of our dataset shape. Such a data has to be standardized and scaled into the same value range in order to minimize dominance of a single feature. For example, speed has values [0,10], and heart rate [100, 200], and yet altitude something between [50, 300]. This may also produce a question of how the data should be standardized. In the study centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. In the figure 2.2C effect of standardization is demonstrated in a single random sequence sample.

When combining interlaced sequences from different types of categories into the same picture, somewhat clear distinctivity and particularity between them can be observed. See the figures 2.2D and 2.2E.

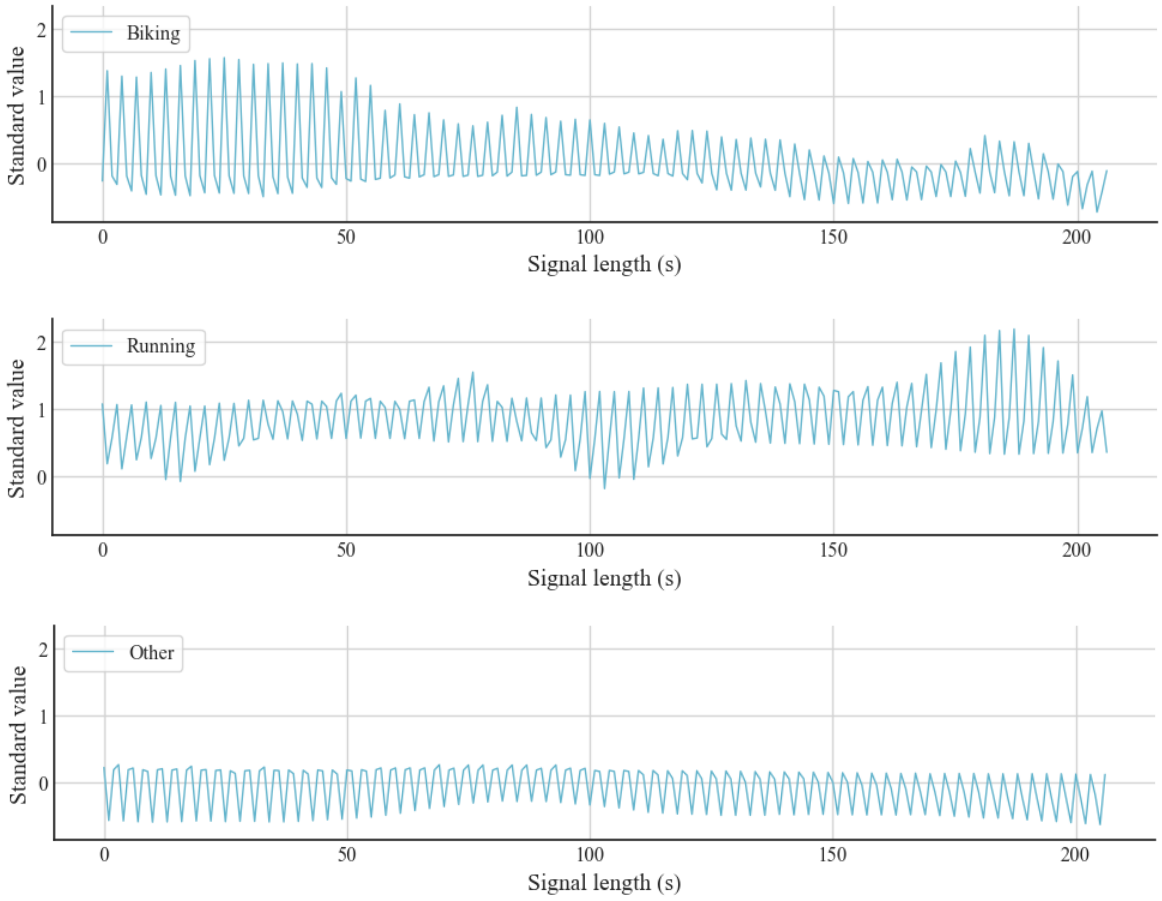


Fig. 2.2D: Standardized sequence samples for three different categories (shared y-axis).

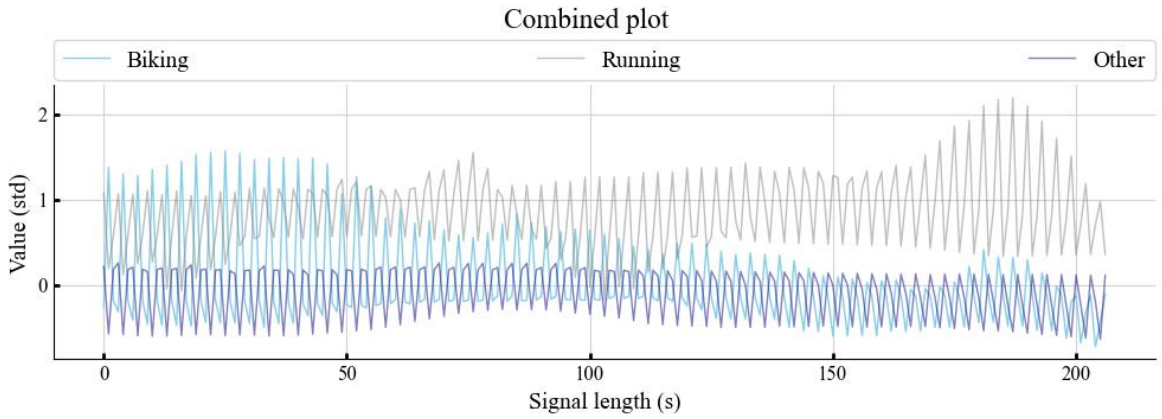


Fig. 2.2E: Combined standardized sequence samples for three different categories. Signals from the figure 2.2D in the same figure (just for signal comparison).

Also, we can visualize same category activities in the same picture in order to identify similar patterns. In figure 2.2F, three random sequences of the running category are visualized.

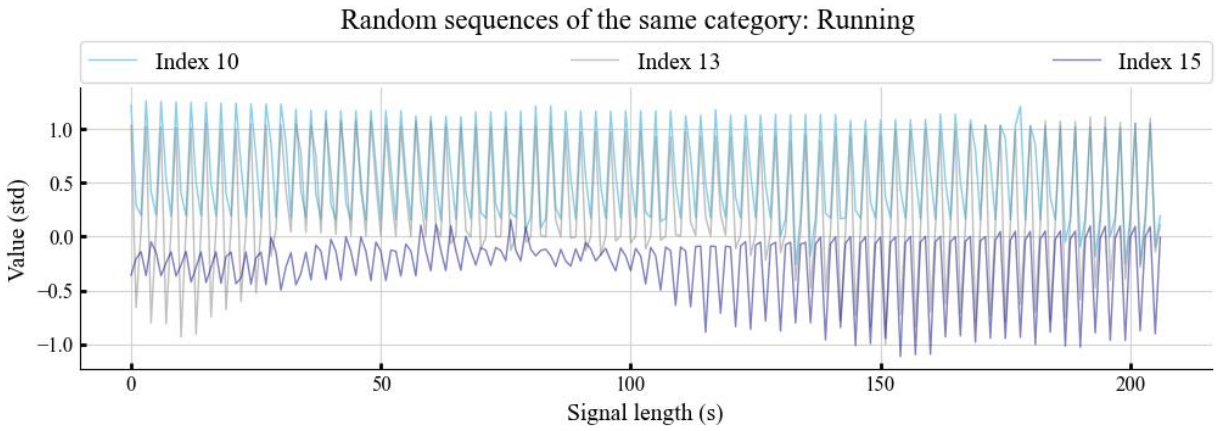


Fig. 2.2F: Three standardized sequences for the category running (random samples/indexes).

Train and test data

Since generated new data structures retain the original sequence order, labels will only be transformed to categorical codes. For train and test data split, we use 80|20 ratio. Data will be shuffled and stratified according to the label values because the dataset is unbalanced as can be seen in figure 2.2G.

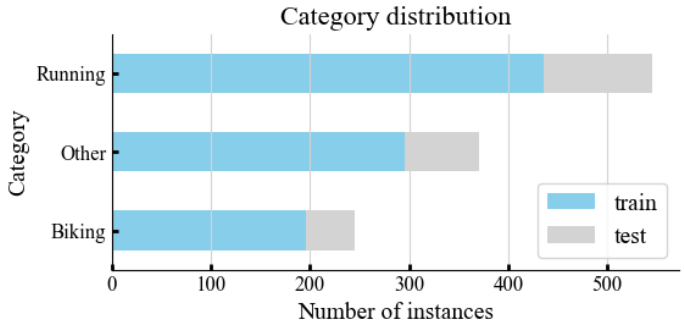


Fig. 2.2G: Category and train-test split distribution.

The final train and test data sizes before classification are depicted in figure 2.2G. It visualizes category distribution in train and test splits in which *other*-category consists of sports walking, skiing, and roller-skiing. In *x*-axis there is a number of instances (segments). Lastly, train data is transformed to the nested table for sktime classifiers, and furthermore, nested table have to be tabularized for sklearn classifiers. From these preconditions we proceed to perform classification analysis for time series, and for some well known classical machine learning models. Yet, it is better to be clear that we have now transformed multivariate data into univariate in order to make data applicable for univariate *sktime* classifiers. In other words, we have now constructed data for univariate time series classification task. Dataframe follows the structure like in the table 2.2C.

Table 2.2C: One-dimensional multivariate signal data structure with indexes [0, m].

	Dimension 0, indexes 0-m									
Label	x_1	y_1	z_1	x_2	y_2	z_2	…	x_m	y_m	z_m
A	ax_1	ay_1	az_1	ax_2	ay_2	az_2	…	ax_m	ay_m	az_m
B	bx_1	by_1	bz_1	bx_2	by_2	bz_2	…	bx_m	by_m	bz_m
C	cx_1	cy_1	cz_1	cx_2	cy_2	cz_2	…	cx_m	cy_m	cz_m
…	…	…	…	…	…	…	…	…	…	…
N	nx_1	ny_1	nz_1	nx_2	ny_2	nz_2	…	nx_m	ny_m	nz_m

In table 2.2C, the label column consists of categories in random order, which means there are no significance or any relation between rows. Column name *Dimension 0* depicts how the data is stored in nested univariate data table. Sub-columns from $\{x_1, y_1, z_1\}$ to $\{x_m, y_m, z_m\}$ demonstrate index values in a single cell in nested data table. This kind of data structure is required by *sktime* univariate TSC models.

Multivariate data structure for M-TSC

For multivariate classification the same pre-processed data as described in the section of dataset construction is used but in the last step features are processed separately. For each feature of *heart rate*, *speed*, and *altitude* we take a transpose of feature vector and then use a built in function to transform column data to a one-dimensional nested structure as discussed previously. Lastly, those individual one-dimensional tables are combined to construct a three-dimensional nested data structure consisting of these features in the format as depicted in table 2.2D.

Table 2.2D: Multi-dimensional nested data structure with labels. Dimensions {x,y,z}.

Index	**Dim $_x$**	**Dim $_y$**	**Dim $_z$**	**Label**
0	$ax_1 : ax_m$	$ay_1 : ay_m$	$az_1 : az_m$	A
1	$bx_1 : bx_m$	$by_1 : by_m$	$bz_1 : bz_m$	B
2	$cx_1 : cx_m$	$cy_1 : cy_m$	$cz_1 : cz_m$	C
…	…	…	…	…
n	$nx_1 : nx_m$	$ny_1 : ny_m$	$nz_1 : nz_m$	N

This data structure is suitable to conduct multivariate time series data classification using built in functions and models from sktime API, such as MUSE. But also, it enables us to implement ensemble models in order to use optimal classifier for each feature and then combine predictions to further enhance classification results.

CLASSIFICATION METHODS

3.1 Standard CML classification

Classification and model selection methods

In this section, we will evaluate the performance of nine S-CML algorithms on the sport activity dataset. Algorithms are selected based on their presence and overall performance on previous HAR studies [1,10,19–21]. In general, selected algorithms have shown good suitability in other corresponding tasks. Selected algorithms are k-Nearest Neighbor (kNN), Decision Tree (DT), Random Forest (RF), Naïve Bayes (G-NB), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Support Vector Machine (SVM) and Logistic Regression (LR) classifiers. We investigate algorithm selection procedure and show preliminary results with default hyperparameters (*python sklearn API*). Then algorithms will be evaluated using both PCA and standardized data. We use grid search cross validation technique to optimize hyperparameters and explain the results. Lastly, brief analysis considering only aspects of algorithm selection and hyperparameter optimization techniques and achieved results will be provided.

Model selection concept

This section considers a certain subset of common S-CML algorithms and evaluates their accuracy in a given dataset. Thus, there is no need for comprehensive and careful algorithm selection procedure and due to the small size of the dataset the most suitable one will be found simply by *brute force* testing. However, to ensure that the selected subset of S-CML algorithms contains the most powerful ones, fundamental premises are reasonable to be taken into account.

Classification problem identification is probably the most significant individual thing when selecting appropriate algorithm for classification. SAC problem is supervised classification case in which algorithms like Logistic Regression, Random Forest, and Decision Tree can be used. In other words, unsupervised classification algorithms, k-Means for instance, which may be more optimal in clustering tasks, will be excluded. The size of the dataset is also an important factor that should be considered while selecting S-CML algorithms. Because the size of the dataset is small, we can select algorithms with *low bias* and *high variance* like Naive Bayes. We may also apply *high bias* and *low variance* algorithms like kNN, and SVM, which basically performs better in larger datasets. CML model training time is sometimes crucial when the dataset is large and therefore affects model selection. Complex algorithms like SVM and RF may assume remarkably time for computation. Yet a

crucial factor in model selection is linearity of the dataset since there are models which are restricted to linear datasets. For example, in principal component analysis of this thesis, and further, in correlation matrix A-2.1C, clear linear relationship between input variables and target variables was detected and thus we can apply linear models with expectation of good results. Also, if the dataset contains unnecessary and irrelevant features, SVM algorithm may suite the best in such cases. [24] Since careful feature selection is performed and we use PCA components, this issue will not be emphasized in this thesis.

Model evaluation is performed using simple standardized data with all the resulting features (21 + target) as implemented in section *2.1 Dataset for Standard CML,* and in PCA data using all the components generated from those 21 features. Data was split into 70% train data and 30% test data parts for the *holdout validation*. The basic idea of the *holdout validation* procedure and dividing data into train and test parts are shown in the appendix A-3.1A.

In the worst-case scenario, the test set may not contain any instance of a minority class at all. Therefore, the dataset was divided in a stratified fashion according to class labels to ensure sufficient number of instances for each of them in train data. Stratification is an approach to maintain the original class proportion in resulting subsets. Random subsampling in non-stratified fashion is usually a big concern when working with relatively small and unbalanced datasets. In addition, shuffled data, and random state parameter with value 24 were used. At first, models were scored with default hyperparameters in standard data using Python *scikit-learn API* library.

Model hyperparameter optimization

In this section we continue the proposed model selection process by tuning model settings. As discussed in previous sections, every machine learning algorithm consists of a large number of settings which need to be specified. Since models are different types, they also need model specific tunable knobs, called hyperparameters. For example, considering the kNN algorithm we need to specify the best integer value of k. Because there exists no trivial way to do that, it must be simply tested with several values and select one which produces the most accurate result. In a model optimization, the method represented in figure 3.1A will be applied. We continue from the hyperparameter tuning by applying the baseline method presented in figure 3.1B in order to find the best model.

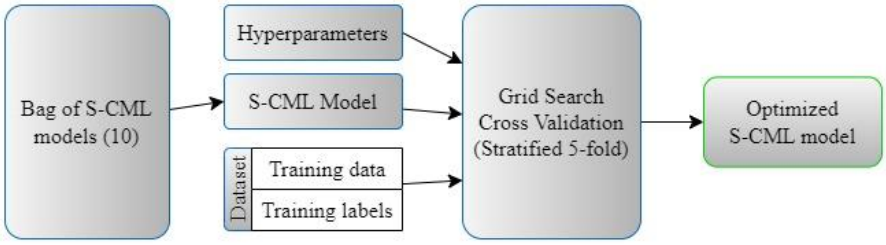


Fig. 3.1A: CML model qualifying method.

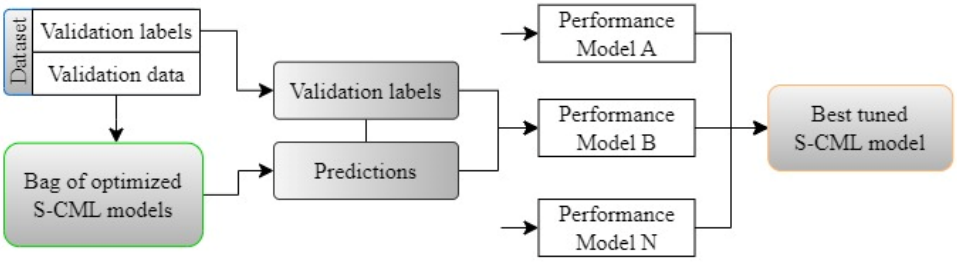


Fig. 3.1B: Best model selection method.

As can be seen from the section *4.1 Preliminary results*, considerable part of the S-CML models provides good results using default hyperparameters set in *python sklearn* library, and there is no reason to expect significant improvement, especially in the models which are already performing over 95% accuracy. Hyperparameter optimization tasks may often require very massive iterative computations which is problematic when dataset is big and high dimensional. However, in this small size dataset hyperparameters will be optimized by executing 20 iterations for each of the model trying to average variance and minimize randomness in results. Grid search *k*-fold cross-validation technique is used, which means that all possible combinations of predefined hyperparameter values were tested. The basic idea of *k*-fold model validation for hyperparameters is depicted in the appendix A-3.1B. In the following are defined metrics for the model validation as a clarification.

Prediction accuracy [24]

Since this report mostly refers to the prediction accuracy instead of the error, we define Kronecker's Delta function:

$$\delta\big(L(\hat{y}_i\,, y_i)\big) = 1 - L(\hat{y}_i\,, y_i)$$

Such that for loss:

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

And for accuracy:

$$\delta\big(L(\hat{y}_i, y_i)\big) = \begin{cases} 0 & if \ \hat{y}_i \neq y_i \\ 1 & if \ \hat{y}_i = y_i \end{cases}$$

$$ACC_S = \frac{1}{n}\sum_{i=1}^{n} \delta\big(L(\hat{y}_i, y_i)\big)$$

Where L represents the 0-1 loss, which is computed from a predicted class label ($\hat{y}_i$) and a true class label (y_i) for $i = 1, \ldots, n$ in dataset S. Also, mean squared error was used in default parameter test, but for technical issues it was not reasonable to use it in hyperparameter optimization:

$$MSE_S = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Repeated Holdout Validation

To obtain a more robust performance estimate that is less variant about how the data is divided into training and test sets the holdout method was repeated k times with different random seeds and then computed the average performance over these *k* repetitions:

$$ACC_{avg} = \frac{1}{k}\sum_{j=1}^{k} ACC_j$$

where ACC_j is the accuracy estimate of the j[th] test set of size m,

$$ACC_j = 1 - \frac{1}{m}\sum_{j=1}^{m} L(\hat{y}_i, y_i)$$

This repeated holdout procedure provides a better estimate of how well S-CML model may perform on a random test set compared to the standard holdout validation method. Also, it provides information about the model's stability and performance fluctuation among different training set splits. [24]

Hyperparameter values

When selecting hyperparameter values, the actual challenge is to determine proper values to be tested because we strive to minimize value range due to computational issues. Also, if a too large step value is used it is possible that certain sensitive and narrow value range is skipped and therefore certain models might be rejected incorrectly. In grid search cross-validation all possible combinations are tested but because test values are selected manually there exists a risk that value range or scale is not correct. However, in order to select a model for further tuning approximate optimal parameters

for each model are obviously found. Tested hyperparameters and best combinations for each model are represented in the table A-3.1C.

When selecting hyperparameters there appears a problem: because hyperparameters are optimized separately in each iteration for a single model with different random state of data split, exact values cannot be chosen unambiguously. Therefore, mode values of hyperparameter combinations are used. In continuous floating point value case, there might be different values equal with number of iterations. In that sense, the optimal hyperparameters represented in the table A-3.1C are indicative only.

3.2 Univariate TSC

U-TSC Models and evaluation metrics

Univariate TSC analysis consists of eight univariate *sktime* classifiers, Time Series Forest Classifier (TSF), Supervised Time Series Forest (STSF), Random Interval Spectral Forest (RISE), Random Interval Classifier (RIC), Shapelet Transform Classifier (ST), k-Nearest Neighbours Time Series Classifier (kNN-TS), Composable Time Series Forest Classifier (CTSF) which is like TSF but has extended composability, and Word Extraction for Time Series Classification (WEASEL). The individual and relative performance of these models will be examined. As we use interlaced univariate signals, which are constructed from multivariate data, classification task is reduced to a univariate TSC. First, we take a brief preview and introduce principles of each U-TSC algorithm and discuss them in terms of *sktime* python library. Then we proceed into actual test results and explore model performance individually using common classification evaluation metrics like overall model *accuracy*, *precision*, *recall*, and *f1-score*. For model comparison, we use some additional metrics such as *Receiver Operating Characteristic* (ROC) curve and its evaluation method *Area Under Curve* (AUC). Also, we consider computation time for model training and testing. Because there occurs a significant fluctuation between consecutive algorithm executions, mainly caused by data shuffling, we conduct iterative classification for each model. That allows us to evaluate model stability using variance and mean values of accuracy.

Sktime API

Sktime is a unified framework for machine learning with time series. It provides an easy to use, flexible, and modular open-source framework for a wide range of time series machine learning tasks. It offers *scikit*-learn compatible interfaces and model composition tools, with the goal of making the ecosystem more usable and interoperable as a whole. *Sktime* features a unified interface for multiple

time series learning tasks. Currently, it supports forecasting, time series classification, time series regression and time series clustering. [25] According to the study of Löning et al. (2019), "the main goal of sktime is to create a unified API for multiple time series tasks, extending the common scikit-learn interface to the temporal setting, while staying close to its syntax and logic whenever possible. Following *scikit-learn API* allows us to re-utilize many of the algorithms available in scikit-learn, which is especially useful because of reduction to tabular tasks and because many specialized algorithms for time series are composites with tabular supervised learning algorithms as their components." Sktime features are represented in the table 3.2A.

Table 3.2A: Sktime features.

	Description
Definition	Modular, principled, and object-oriented API for machine learning with time series, for the purpose of specifying, fitting, applying, and validating machine learning models.
Interpreter	Interactive Python interpreter. No command-line interface or graphical user interface
Syntax	Interactive user experience with scikit-learn like syntax conventions.
Computation	In-memory computation of a single machine. No distributed computing
Data handling	Medium-sized data in Pandas and NumPy
Status	Early stage of development (2019) and currently includes several state-of-the-art algorithms for time series classification.

Sktime algorithms are divided into sub-categories which are *interval based*, *distance based*, *shapelet based*, *dictionary based, feature based*, and *deep learning.* [25,26] From these categories, models were selected as follows: four interval-based models TSF, STSF, CTSF, and RISE. Model RIC is feature based, STC is shapelet based, and kNN-TS is distance based. Lastly, WEASEL was selected to represent dictionary based models in our U-TSC study. The majority of applied U-TSC models are based on a decision tree algorithm and uses Random Forest (RF) as a base model, whose principals are depicted in figure 3.2A. In the next section, these selected models are introduced. Also, an incomplete set of model pseudocodes are available as appendix A-3.2A.

Random Forest algorithm

Random Forest is a popular machine learning algorithm used for several types of classification tasks. It is an ensemble of tree-structured classifiers. Every tree of the forest gives a unit vote, assigning each input to the most probable class label. It is a fast method and robust to noise, which can identify non-linear patterns in the data. RF can easily handle both numerical and categorical data. One

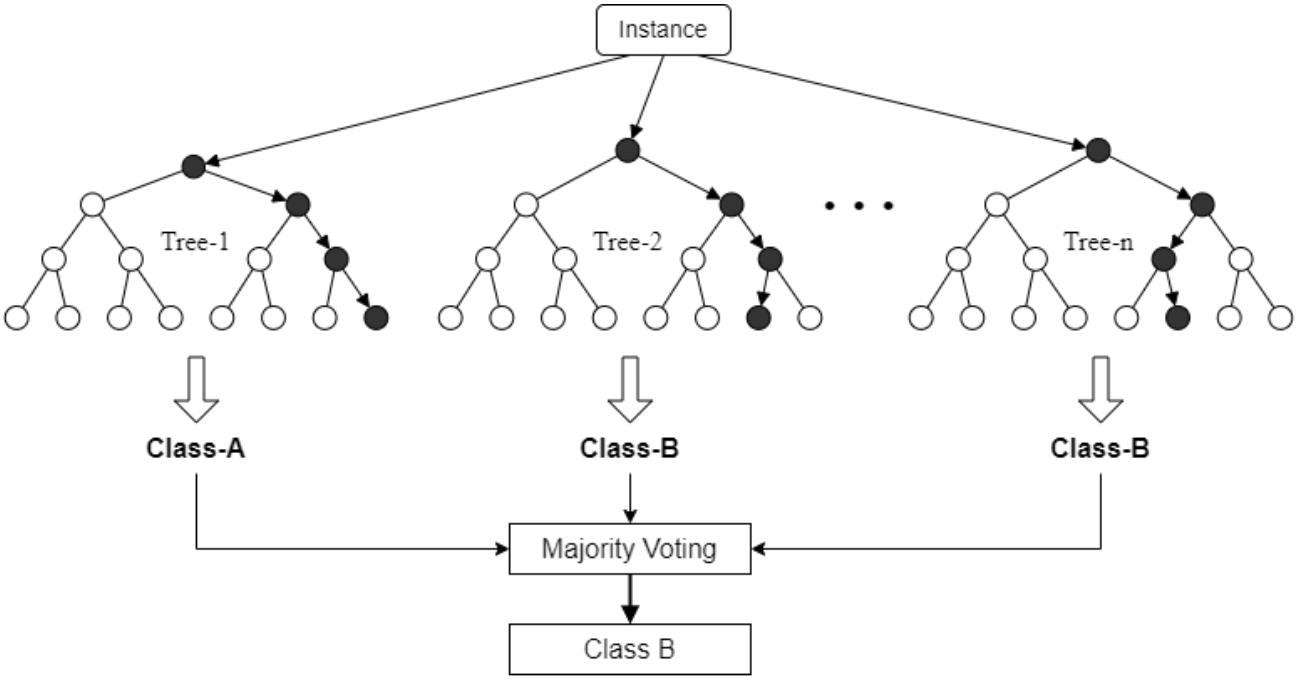


Fig. 3.2A: Principals of Random Forest algorithm.

of the major advantages of RF is that it does not suffer from overfitting, even if more trees are appended to the forest. [27]

Time Series Forest Classifier (TSF)

TSF employs a combination of entropy gain and a distance measure, referred to as the Entrance (entropy and distance) gain, for evaluating the splits. TSF randomly samples features at each tree node and has computational complexity linear in the length of time series and can be built using parallel computing techniques. The temporal importance curve captures the temporal characteristics useful for classification. Using simple features such as mean, standard deviation and slope it is computationally efficient and outperforms strong competitors such as 1-Nearest Neighbours with dynamic time warping. [28] TSF classifier adapts the *Random Forest* classifier to series by steps below:

Table 3.2B: TSF algorithm steps.

Step	Description
1	Split the series into random intervals, with random start positions and random lengths
2	Extract summary features (mean, standard deviation, and slope) from each interval into a single feature vector.
3	Train a decision tree on the extracted features.
4	Repeat steps 1–3 until the required number of trees have been built or time runs out.

Supervised Time Series Forest Classifier (STSF)

STSF is an ensemble of decision trees built on intervals selected through a supervised process using input of n series in length m for each tree. Then it samples data using class balanced bagging. STSF improves the classification efficiency by examining only a set of sub-series of the original time series, and its tree-based structure allows interpretable outcomes. STSF adapts a top-down approach to search for relevant sub-series in three different time series representations prior to training any tree classifier, where the relevance of a sub-series is measured by feature ranking metrics, such as supervision signals. [29]

Table 3.2C: STSF algorithm steps.

Step	Description
1	Input *n* series of length *m* for each tree
2	Sample *X* using class-balanced bagging.
3	Sample intervals for all 3 representations and 7 features using supervised method.
4	Find *mean*, *median*, *std*, *slope*, *iqr*, *min* and *max* using their corresponding interval for each representation, concatenate to form new data set.
5	Build decision tree on new data set.
6	Ensemble the trees with averaged probability estimates.

Random Interval Spectral Ensemble (RISE)

The Random Interval Spectral Ensemble (RISE) is a recently introduced tree-based ensemble time series classification algorithm, where each tree is built on a distinct set of Fourier, autocorrelation, and partial autocorrelation features [30]. RISE is a popular variant of *Time Series Forest.* RISE differs from Time Series Forest in two ways. First, it uses a single time series interval per tree. Second, it is trained using spectral features extracted from the series, instead of summary statistics. RISE uses several series-to-series feature extraction transformers, including fitted auto-regressive coefficients, estimated autocorrelation coefficients, and power spectrum coefficients (the coefficients of the Fourier transform). Class probabilities are calculated as a proportion of base classifier votes. RISE controls the run time by creating an adaptive model of the time to build a single tree. This is important for long series, such as audio, where very large intervals can mean very few trees. [31] The RISE algorithm is straightforward containing the steps described in table 3.2D.

Table 3.2D: RISE algorithm steps.

Step	Description
1	Select random interval of a series (length is a power of 2). For the first tree, use the whole series.
2	For the same interval on each series, apply the series-to-series feature extraction transformers (autoregressive coefficients, autocorrelation coefficients, and power spectrum coefficients)
3	Form a new training set by concatenating the extracted features.
4	Train a decision tree classifier
5	Ensemble 1–4.

Random Interval Classifier (RIC)

This classifier simply transforms the input data using the *random intervals* transformer and builds a provided estimator using the transformed data. Applied estimator defaults to a Rotation Forest with 200 trees. As described in the study of Rodriguez et al (2013), Rotation Forest constructs a forest of trees built on random portions of the data transformed using PCA. [32]

Shapelet Transform Classifier (STC)

Shapelets are subsequences, or small sub-shapes of time series that are representative of a class. They can be used to detect phase-independent localized similarity between series within the same class. Shapelet-based classifiers search for shapelets with discriminatory power. These shapelet features can then be used to interpret a shapelet-based classifier — the presence of certain shapelets make one class more likely than another.

K-Nearest Neighbors Time Series (kNN-TS)

KNN-TS classifier finds the k nearest neighbors of a time series and the predicted class is determined with *majority voting*. A key parameter of this algorithm is the *metric* used to find the nearest neighbors. A popular metric for time series is the Dynamic Time Warping metric.

Composable Time Series Forest (CTSF)

CTSF is like Time Series Forest Classifier as described. A time series forest is an adaptation of the random forest for time-series data. It fits a number of decision tree classifiers on various sub-samples of a transformed dataset and uses averaging to improve the *predictive accuracy* and control *over-fitting*.

Word Extraction for Time Series Classification (WEASEL)

The WEASEL is a fast and accurate univariate TSC model and was first presented in the study of Schäfer and Leser (2017). It applies a supervised symbolic representation to transform subsequences to words, uses statistical feature selection, and subsequently feeds the words into a logistic regression classifier. WEASEL is among the most accurate and fastest univariate TSC. The advantage of WEASEL lies in its specific method for deriving features, resulting in a significantly smaller and more discriminative feature set. Schäfer and Leser found WEASEL more accurate than the best non-ensemble algorithms at orders-of-magnitude lower classification and training times on the popular UCR benchmark of 85 TS datasets. Furthermore, the model achieved almost as accurate results as ensemble classifiers, whose computational complexity makes them inapplicable even for mid-size datasets. [33]

HIVE-COTEv1.0

On their extensive multivariate TSC classification bake off, Ruiz and others (2021) stated that one of the most straightforward techniques to adapt univariate TSC algorithms to multivariate is to ensemble models which are built on each dimension independently. Ensemble models consider dimensions as independent and ignore relations among them. This approach is a good baseline for assessing and contrasting bespoke M-TSC classifiers which can also model dimension dependencies. According to them, one of the most accurate approaches to U-TSC is the Hierarchical Vote Collective of Transformation based Ensembles (HIVE-COTE). HIVE-COTE is a heterogeneous meta ensemble for time series classification. The key principle behind HIVE-COTE is that time series classification problems are best approached by careful consideration of the data representation, and that with no expert knowledge to the contrary, the most accurate algorithm design is to ensemble classifiers built on different representations. [34]

The latest version, HIVE-COTEv1.0 or referred simply as HIVE-COTE, combines STC, TSF, RISE, and Contractable Bag of Symbolic-Fourier Approximation Symbols (CBOSS) – that is not included in U-TSC of this thesis separately – using a weighted probabilistic ensemble. The simplest way to build a multivariate HIVE-COTE is to build each component as an independent ensemble, then to combine the components in the usual way. To clarify, each component builds a separate classifier on every dimension, then combines the predictions from each dimension to produce a single probability distribution for each of STC, TSF, CBOSS and RISE. [35].

HIVE-COTE adopts the Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE) ensemble structure, summarized in pseudocode (A-3.2A). CAWPE uses an estimate of the

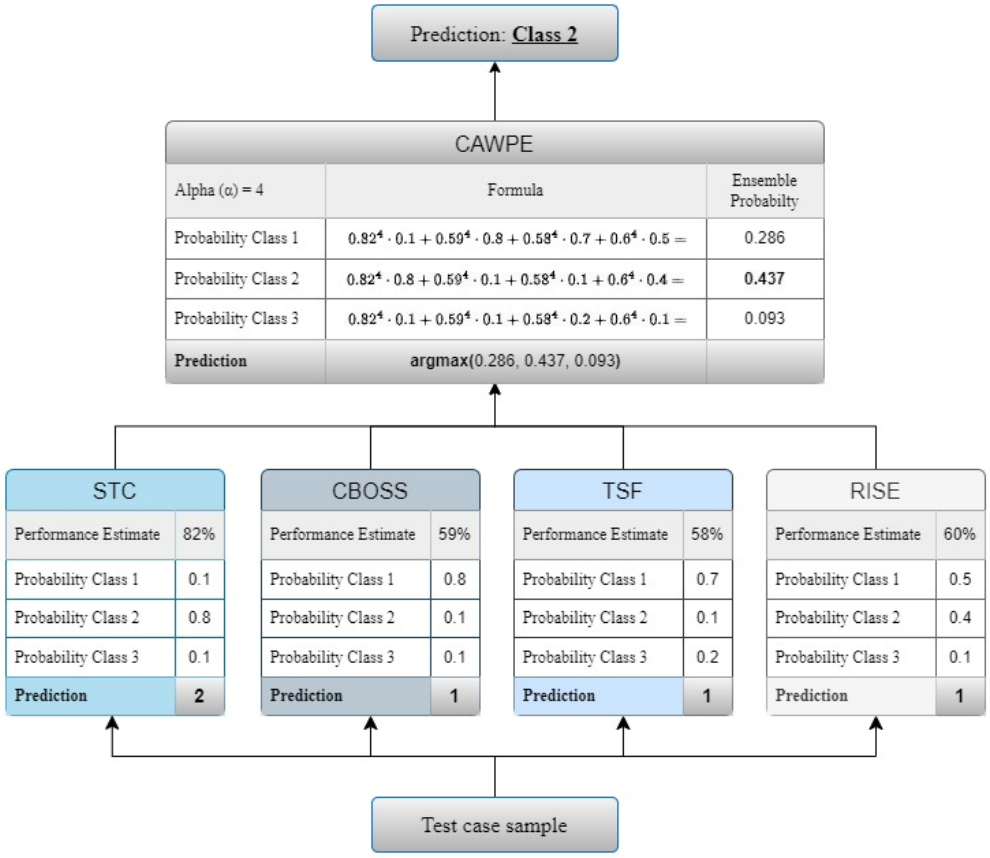


Fig. 3.2B: An overview of the ensemble structure of HIVE-COTEv1.0. Each module produces an estimate of the probability of membership of each class. The control unit (CAWPE) combines these probabilities, weighted by an estimate of the quality of the module found on the train data. (Bagnall et al., 2020)

accuracy of each classifier to weight the probability estimates of each component. It constructs a tilted distribution through exponentiation (using α) to extenuate differences in classifiers. The weight for each component is found either through ten-fold cross validation, or, if the classifier has the capability to estimate its own performance, internally. [34] The very exhaustive way to present HIVE-COTEv1.0 algorithm is to use a flowchart, as in figure 3.2B.

Model hyperparameters

Due to a limited number of computational resources in order to conduct comprehensive TSC hyperparameter optimization we decided to use default setup of *sktime API* for python. Secondly, considering an approach used in the study, which is to use in uncontrolled environment created dataset including very likely 2-5% wrong labeled activities, especially confusion between walking and running, having great variance in model performance with same setup depending on train-test data division, we did not see any reason to implement model fine tuning in this context and dataset. As we

basically estimate about 1-2 percent improvement through optimized hyperparameters, there is no reason to expect statistical significance in model performance with an observed accuracy variance, and achieved advantage could be difficult to interpret. Thirdly, as we include a lot of models and methods in the study, model hyperparameters can be focused on future studies using a smaller set of classifiers with better quality and size dataset. Adopted model setup can be found in the appendix A3.2B

3.3 Multivariate TSC

A time series is a series of observations over a period of time. When the observed space is multidimensional, the time series becomes multivariate. Multivariate Time Series Classification (M-TSC) is a supervised learning problem in which each sample consists of one or more time series attributes.

WEASEL-MUSE

For multivariate TSC we use Multivariate Unsupervised Symbols and dErivatives (MUSE) algorithm, also known as WEASEL+MUSE, which is implementation of multivariate version of WEASEL (Word ExtrAction for time SEries cLassification) and referred to as just MUSE. WEASEL+MUSE is a multivariate dictionary classifier that builds a bag-of-patterns (BOP) using Symbolic Fourier Approximation (SFA) for different window lengths and learns a logistic regression classifier on this bag. [36]

The novelty of WEASEL+MUSE lies in its specific way of extracting and filtering multivariate features from MTS by encoding context information into each feature. It uses statistical feature selection, derivatives, variable window lengths, bi-grams, and a symbolic representation for generating discriminative words. WEASEL+MUSE provides tolerance to noise due to use of the truncated Fourier transform, phase invariance, and superfluous data/dimensions. Thereby, WEASEL+MUSE assigns high weights to characteristic, local and global substructures along dimensions of a multivariate time series. In our evaluation on altogether 21 datasets, WEASEL+MUSE is consistently among the most accurate classifiers and outperforms state-of-the-art similarity measures or shapelet-based approaches. It performs well even for small-sized datasets, where deep learning based approaches typically tend to perform poorly. When looking into application domains, it is best for sensor readings, followed by speech, motion and handwriting recognition tasks. [36]

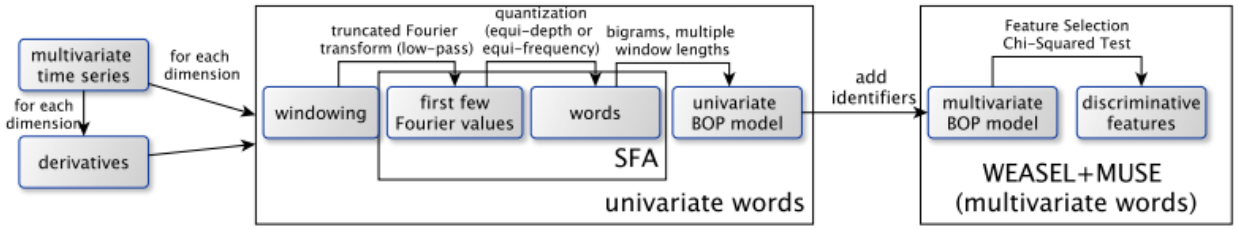


Fig. 3.3A: WEASEL+MUSE Pipeline: Feature extraction, univariate Bag-of-Patterns (BOP) models and WEASEL+MUSE. [36]

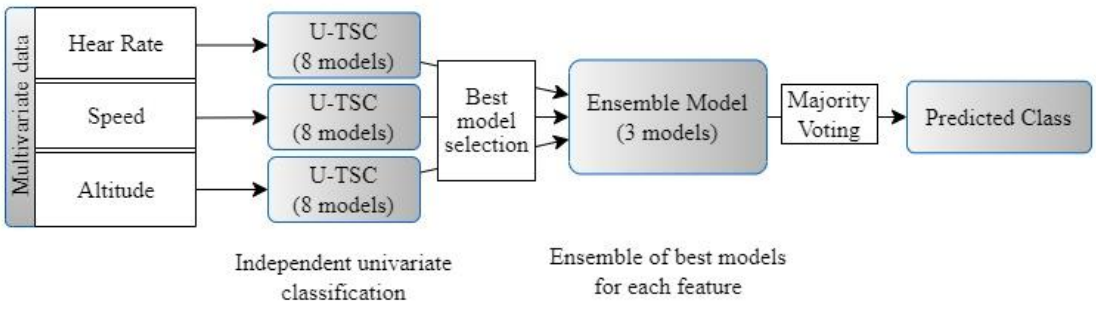


Fig. 3.3B: Column ensemble model pipeline for multivariate data.

Column Ensemble

In addition to MUSE, we implement classification for each of the three dimensions in the dataset separately using selected univariate models from the chapter *3.2 Univariate TSC* and then select the best models in terms of accuracy, ROC-AUC value, and algorithm execution time. Then we conduct a Column Ensemble (CE) in multivariate data from those best performing models in order to further improve classification accuracy and will utilize *sktime API's* built-in *ColumnEnsembleClassifier* -function. It allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be ensembled to form a single output. Thus, unlike MUSE, the built ensemble model ignores feature correlation, and therefore, uses less information in classification. Method pipeline is presented in figure 3.3B.

3.4 Project implementation

The project code for the classification task was implemented using local Visual Studio Code desktop application and its Jupyter-notebook extension for Python programming language. File execution time for TSC including data preparations, and actual model fitting with predictions for all the

models required 4 hours and 50 minutes. Standard CML models needed about 70 minutes with hyperparameter optimization for the particular parameters depicted in the appendix A-3.1C. Project code was divided into three parts: first part for S-CML classification, and second part for U-TSC and M-TSC, and third for data visualizations and analysis. Thus, classification data was first stored into a comma separated value (csv) files to be easily accessible in later analysis. For the classification executions itself we used a specific functions wherein models were iterated over a nested loop, for example, see the function code for univariate model TSC in A-3.4A. The whole project source code is publicly available on GitHub.com. Find the links in the appendix A-PC.

RESULTS

4.1 Standard CML classification

Preliminary results

Figure 4.1A-a indicates obtained average accuracy results among 20 iterations for each model. Test was implemented by fitting the data to the model and calling score function of that model which calculates R2 (Coefficient of Determination) accuracy value, and it is defined as:

$(1 - u/v)$, where $u = \left(\left(y_{true} - y_{pred}\right)^2\right).sum()$, and $v = ((y_{true} - y_{true}.sum())^2).sum()$

The best possible score of R2 is 1.0 and it can be negative as well if the model is performing weakly. A model that outputs constant prediction for each input will have a score of 0.0. Also, training and scoring times were evaluated with the following results: neural network based classifier MLP was the slowest as expected and performed in average time of 0.27 seconds. Whereas the slowest S-CML model was Random Forest classifier with an approximate total time of 0.26 seconds. Other CML models performed significantly faster using only fractions of consumed time of MLP or RF classifiers. Except LR, all other S-CML models performed quite equally without statistical significance, as can be seen on figure 4.1A-b. KNN is an exceptional model requiring more time for test than training part of the classification. Confusion matrixes for standard data are available as an appendix A-4.1A. but keep in mind, that those results do not correspond with these results presented in figure 4.1A because single model execution results depend on a selected random state value when creating training and test data division.

For PCA data the same test with default model hyperparameters was executed. In general, the usage of principal component data decreased model accuracies (DT, G-NB, RF and MLP), and only

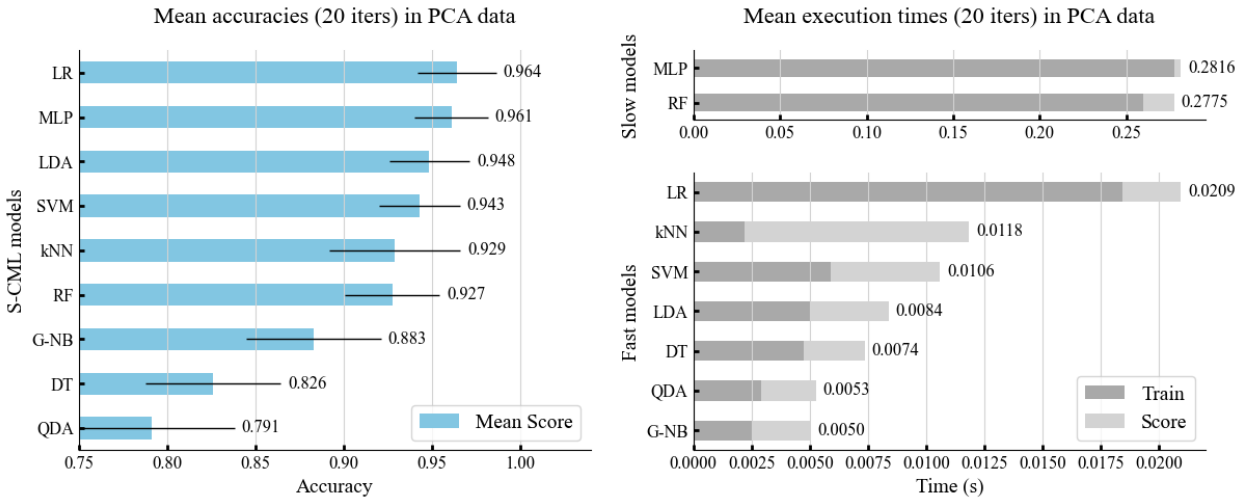


Fig. 4.1A: S-CML model performance statistics in standard data a) On the left, accuracy score values b) On the right, train+scoring times divided into two parts due to scale difference.

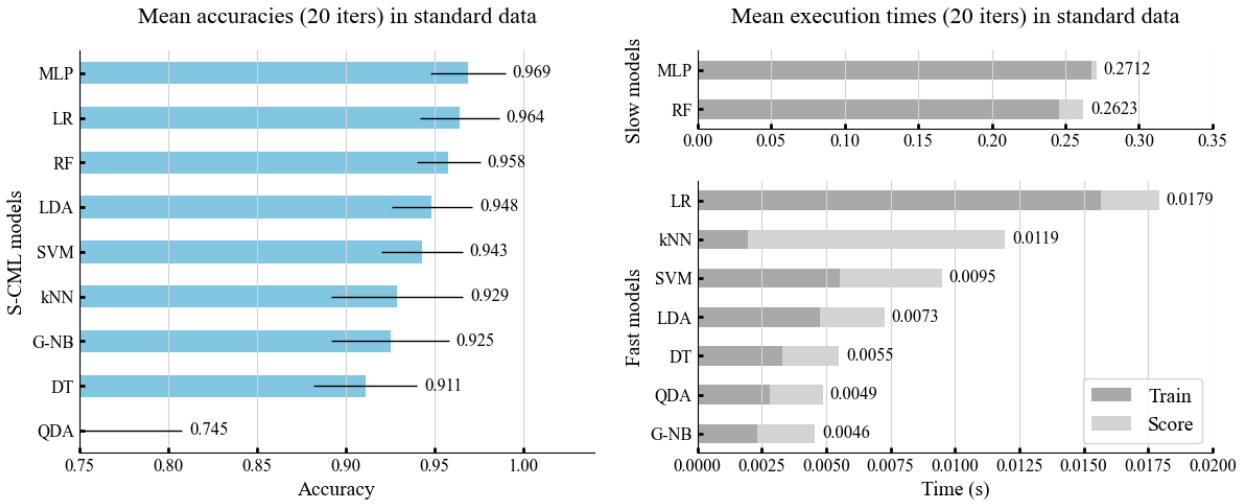


Fig. 4.1B: S-CML model performance statistics in PCA data a) On the left, accuracy score values b) On the right, train+scoring times divided into two parts due to scale difference.

QDA model performance improved. In other models, changes were not obtained. Similarly, as in the standard data, MLP and LR were two best performing models by 2% marginal, but also among RF having the greatest time complexity. See figure 4.1B. As we used all the PCA components without dimensionality reduction due to its non-existing significance in this particular small size dataset, data amount was the same, and therefore model execution times.

At this point, we have quite comprehensive overall insight about how different types of S-CML models will perform at the minimum. Also, we may suppose that the adopted dataset is of sufficient quality because several models scored near or over 95%. The results are promising before hyperparameter optimization, and we may expect very good results after model tuning. Further, it is reasonable

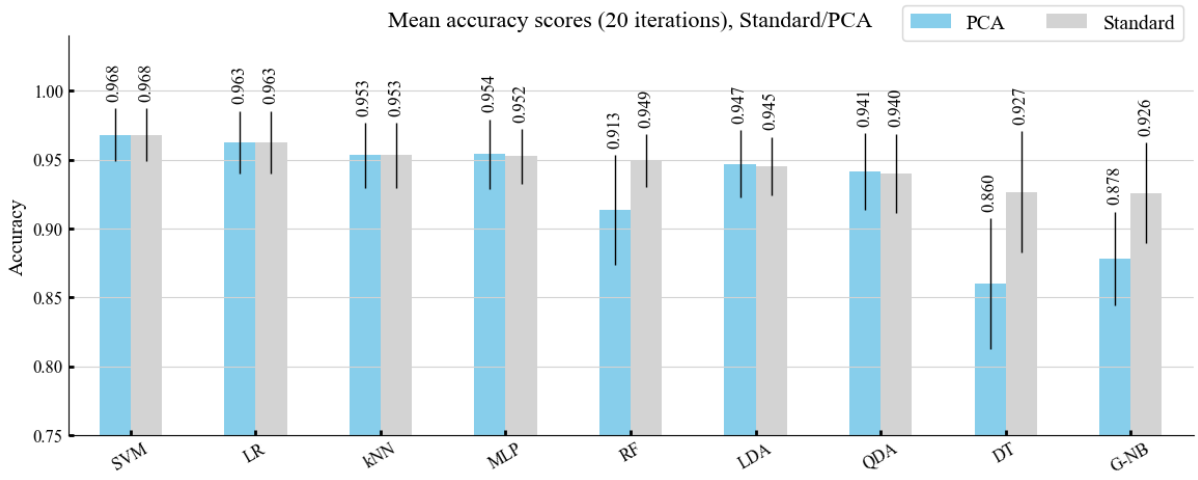


Fig. 4.1C: Accuracy score values with optimal hyperparameters in Standard and PCA data.

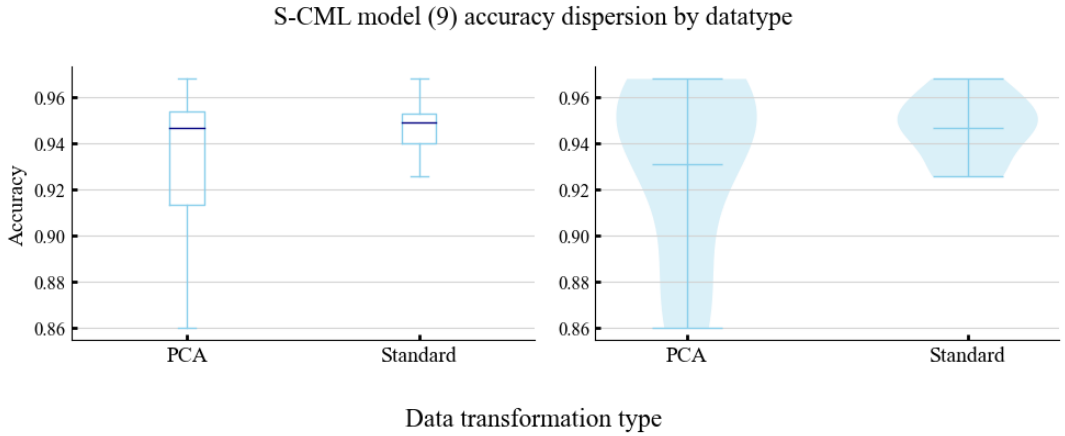


Fig. 4.1D: Accuracy dispersion among data types with optimized hyperparameters (8 S-CML models + MLP).

to keep in mind that single activities might still be misclassified by the user, that will make faultless classification impossible. Referring to figures 4.1A and 4.1B, there are violin plot graphics available in appendix A-4.1B. A violin plot is more informative than a plain box plot. While a box plot only shows summary statistics such as mean/median and interquartile ranges, the violin plot shows the full distribution of the data. Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator.

Prediction results with optimized hyperparameters

Similarly, as for the models with default parameters, classification for fine-tuned S-CML models were implemented using 20 iterations in order to compensate problematic small dataset, and mean accuracy values used to rank models. In general, better accuracies were obtained in standard data as can be seen in figure 4.1C. All the models allowed over 92 percent accuracy which emphasizes the

Table 4.1: S-CML classification results (20 iterations) with optimized hyperparameters in standard data. Notice, that one test sample represents 1,6% of accuracy since test data has only 64 instances. Therefore, accuracy 0.984 means 1 error etc.

Model	Low	High	StD	Mean	Train (ms)	Test (ms)	Total (ms)
SVM	0.938	1	0.019	**0.968**	4.51	2.78	7.29
LR	0.906	1	0.023	**0.963**	20.86	2.66	23.52
kNN	0.906	0.984	0.024	**0.953**	3.96	10.01	14.04
MLP	0.922	1	0.020	**0.952**	212.12	3.51	215.63
RF	0.906	0.984	0.019	**0.949**	51.80	6.77	58.57
LDA	0.906	0.984	0.021	**0.945**	5.90	3.02	8.92
QDA	0.859	0.984	0.029	**0.940**	3.79	2.67	6.46
DT	0.844	1	0.044	**0.927**	4.19	4.00	8.20
G-NB	0.859	0.984	0.037	**0.926**	3.04	2.39	5.43

importance of finding optimal hyperparameter values for certain models. However, in many cases accuracy might be even worse when inspecting results, but this is caused by shuffling train/test division which makes these results not directly comparable with classification results when using default ones. SVM and kNN algorithms improved ~2,5%, but QDA even 15% in principal component data and ~20% in standard data. Thus, we achieved the best results by SVM model, but also found that at least 7 out of 9 models have potential to be considered in sport activity classification using this type of dataset. Confusion matrixes of a single classification with optimal hyperparameters are available as an appendix A-4.1C. From the accuracy dispersion between standard and PCA data, in figure 4.1D, can be concluded that principal component data produces poorer accuracies on average among all machine learning models, whereas in standard data results are better in general. Two models, DT and G-NB performed significantly poorer in PCA data, which causes highly diverged boxplot compared to standard.

Table 4.1 depicts obtained results with optimal parameters in a numeric format. Models are ordered according to the mean score value. Mean score values are mean values among 20 iterative accuracy scores produced by best hyperparameters. The fact how the data is divided into train and test parts affected the obtained results significantly. In certain test data many of the models were able to produce perfect classification that can be seen by inspecting best accuracy results. Accuracy dispersion was ~6-16%, SVM producing the smallest, and G-NB the largest.

The training and scoring times of the models did not change statistically significantly in proportion compared to defaults as analyzed previously. There occurred a great fluctuation between dif-

ferent consecutive classification executions. Therefore, the final model order by accuracy is only indicative, and instead of focusing on a certain model we are more interested about the general model performance. Because all the models were able to classify activities with only one or zero mistakes, we suppose the dataset consisted of a couple of particular problematic, probably mislabeled activities.

In a decision boundary analysis, it could be possible to identify those single cases and better visualize the sticking points in model wise decision boundary definitions. In this thesis we skip that part because the purpose is not to make deeper level analysis but rather to conduct extensive mapping of the performance of CML methods in small datasets for personalized sport activity classification.

4.2 Univariate TSC

U-TSC analysis introduction

This and the following *single model analysis* sections contain classification reports of all U-TSC models for a single execution of the algorithms. Accuracy scores may vary ±2%, for particular models even more depending on which sequences are in the train and test data, since they are randomly distributed. Model accuracy evaluation is made using *precision*, *recall*, *f1-score*, and *support* metrics, complemented by macro and weighted average, formulated as follows:

$$Precision_c = \frac{TP}{TP + FP}, \qquad Recall_c = \frac{TP}{TP + FN}, \qquad F1score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Main purpose of these separate single executions of the algorithms is to provide more detailed classification analysis data about how algorithms perform precisely with a single activity type separately by using above introduced metrics to measure classification performance. These we cannot benefit as effectively when we complete more iterations and then use mean values of metrics. Therefore, we do not either conduct model comparison but strive to get more accurate approximate understanding about general model behavior.

Classification reports

Classification report tables 4.2A-I for U-TSC, also 4.3A for M-TSC, with confusion matrixes are the results of function execution with the same data setup for each model, using 20% validation data ratio and some certain random state value when shuffling and stratifying segments. They indicate model performance not only as a summary but among categories using traditional classification analysis metrics. Three models, STSF, RIC and WEASEL resulted in an accuracy of 91.8%. Yet, to point out a couple of the most significant aspects, STSF did not make mistakes when setting a label of *other*-type category, referring to a precision value. In general, running activities were classified as the best

success in terms of recall, visualized by darker color in confusion matrixes. Otherwise, there seems to be no easily identifiable similarities or anomalies which could be stated as common factors for all the models in the data. In other words, the performance of the U-TSC models is thus focused on slightly different areas, but actual unambiguous classification problem for some set of individual segment cases or certain area cannot be identified from the data itself, since almost all of them seem to be solvable by some model. This can be suggested based on the classification reports.

Table 4.2A: Time Series Forest (TSF)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.918	0.856	0.939	0.904	0.895
Recall	0.918	0.927	0.824	0.890	0.892
F1-Score	0.918	0.890	0.878	0.895	0.892
Support	49	109	74	232	232
Accuracy	**0.892**				

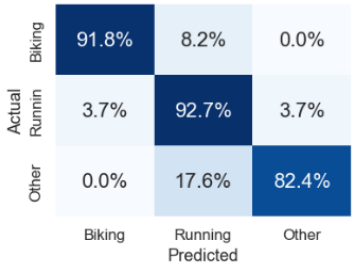


Table 4.2B: Supervised Time Series Forest (STSF)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.865	0.897	1.0	0.921	0.923
Recall	0.918	0.963	0.851	0.911	0.918
F1-Score	0.891	0.930	0.918	0.913	0.918
Support	49	109	74	232	232
Accuracy	**0.918**				

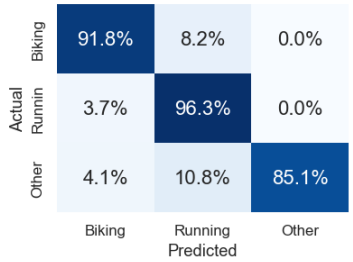


Table 4.2C: Random Interval Spectral Ensemble (RISE)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.902	0.892	0.930	0.908	0.906
Recall	0.755	0.982	0.892	0.876	0.905
F1-Score	0.822	0.935	0.910	0.889	0.903
Support	49	109	74	232	232
Accuracy	**0.905**				

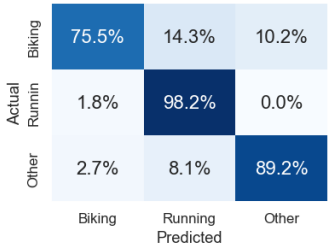


Table 4.2D: Random Interval Classifier (RIC)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.880	0.896	0.985	0.920	0.920
Recall	0.898	0.945	0.892	0.912	0.918
F1-Score	0.889	0.920	0.936	0.915	0.918
Support	49	109	74	232	232
Accuracy	**0.918**				

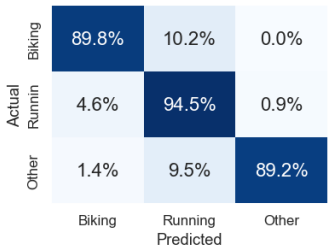


Table 4.2E: Shapelet Transform Classifier (STC)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.857	0.842	0.952	0.884	0.880
Recall	0.857	0.927	0.811	0.865	0.875
F1-Score	0.857	0.882	0.876	0.872	0.875
Support	49	109	74	232	232
Accuracy	**0.875**				

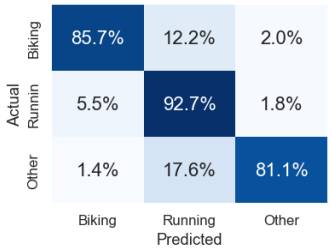


Table 4.2F: k-Nearest Neighbours Time Series (kNN-TS)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.706	0.875	0.710	0.764	0.787
Recall	0.735	0.706	0.892	0.778	0.772
F1-Score	0.720	0.782	0.790	0.764	0.772
Support	49	109	74	232	232
Accuracy	**0.772**				

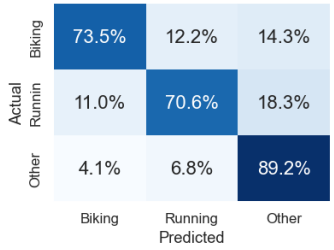


Table 4.2G: Composable Time Series Forest (CTSF)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.900	0.833	0.936	0.890	0.880
Recall	0.918	0.917	0.784	0.873	0.875
F1-Score	0.909	0.873	0.853	0.879	0.874
Support	49	109	74	232	232
Accuracy	**0.875**				

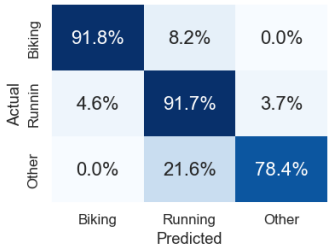
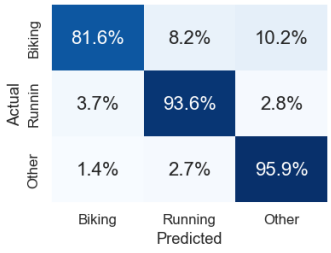


Table 4.2H: Word Extraction for Time Series Classification (WEASEL)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.889	0.944	0.899	0.911	0.918
Recall	0.816	0.936	0.960	0.904	0.918
F1-Score	0.851	0.940	0.928	0.906	0.918
Support	49	109	74	232	232
Accuracy	**0.918**				

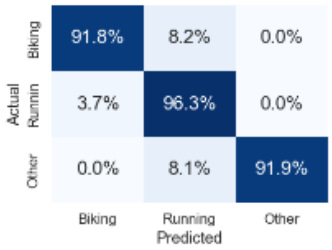


U-TSC ensemble model

Lastly, we implemented HIVE-COTEv1.0 algorithm which ensembles four models, STC, CBOSS, TSF, and RISE as discussed. In single model analysis, from these ensembled algorithms CBOSS was not included for its time consuming nature, and because of limited computational resources in conducting this relatively comprehensive U-TSC. However, it was one of the algorithms ensembled by HIVE-COTE for which one successful execution was completed. As expected, HIVE-COTE allows to improve classification accuracy compared to single models but using an order of magnitude more time. In our test laptop (A-TE), it took 900 minutes (15h) to complete this algorithm with an enhanced accuracy of ~94%, which is about 2 percent higher compared to other U-TSC models.

Table 4.2I: Hierarchical Vote Collective of Transformation based Ensembles (HIVE-COTE)

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.918	0.913	1.0	0.944	0.942
Recall	0.918	0.963	0.919	0.934	0.940
F1-Score	0.918	0.938	0.960	0.938	0.940
Support	49	109	74	232	232
Accuracy	**0.940**				



Because of mentioned environmental limitations, HIVE-COTE is not included in further analysis and model comparison, but it creates a good reference point when analyzing performance of other models, and probably for future studies in SAC field as well.

Classification statistics

In this section we do model performance comparison and also consider accuracy variance and mean score among three algorithm executions. Three iterations was selected for two main reasons: limited computation capacity in our environment (A-TE) to conduct more than three in terms of time, and on the other hand, it seemed to be sufficient in order to reach statistical significance in terms of accuracy variance. However, as there occurs approximately 2% variance caused by the reason of random train/test data distribution as discussed previously, models cannot be ordered by accuracy if using less than three iterations. Thus, in the result table 4.2J, statistics are acquired from three consecutive model executions. Accuracy variance, and mean score are computed from three scores, whereas computational information *train(t)* and *test(t)*, *f1*, and *ROC-AUC* are each from the best scored execution of the classifier. Results are ordered according to the best scores.

Table 4.2J: U-TSC results (3 algorithm executions)

Classifier	Train (t) s	Test (t) s	F1	ROC-AUC	Best Score	Variance	Mean Score
STSF	108	6.0	0.931	0.990	**0.931**	0.004	0.925
RIC	588	40.8	0.927	0.985	**0.927**	0.006	0.920
WEASEL	140	11.4	0.918	0.981	**0.918**	0.000	0.918
RISE	244	38.1	0.905	0.980	**0.905**	0.002	0.902
TSF	22	2.8	0.901	0.979	**0.901**	0.002	0.899
STC	1212	78.7	0.897	0.973	**0.897**	0.004	0.892
CTSF	1692	182.4	0.892	0.978	**0.892**	0.002	0.899
kNN-TS	6	86.9	0.772	0.829	**0.772**	0.000	0.772

Table 4.2K: Prediction results for eight most misclassified segments. Correct predictions are higlighted by grey color. First column indicates specific segment index in the test data (total of 232).

Test data index (232)	TSF	STSF	RISE	RIC	STC	kNN-TS	CTSF	WEASEL	Correct	Errors
220	1	1	1	1	1	1	1	1	**2**	**8**
62	0	0	0	0	0	0	0	0	**1**	**8**
71	0	0	0	0	0	0	0	0	**1**	**8**
97	1	1	1	1	1	1	1	0	**0**	**7**
17	1	0	1	1	1	1	2	1	**0**	**7**
208	1	1	1	1	1	2	1	0	**0**	**7**
138	1	1	0	2	1	1	1	2	**2**	**6**
195	1	1	1	1	1	2	1	2	**2**	**6**
Errors	**8**	**7**	**8**	**7**	**8**	**7**	**8**	**4**		**57**

Misclassification analysis

In misclassification analysis we use actual stored classification results from the models, then combine these results into same table and compare them to actual labels. Thus, we can specify actual segments which have been misclassified, and compare the correct label to predicted one. Table 4.2K contains 8 segments from the test data which has been the most misclassified. Eight errors in the last column mean that all models (8/8) have failed in classification. For instance, the first row of the table in the original test data index 220 of total 232 has been predicted to be class 1, but correct label is 2. Also, we may suggest, that results are biased toward category 1, since signals in indexes [220, 97, 17, 208, 138, 195] are mostly predicted as 1, whereas actual labels correspond to 0 (3) and 2 (3). Also, activities with label 1 have been classified as 0 by all the models in indexes [62, 71].

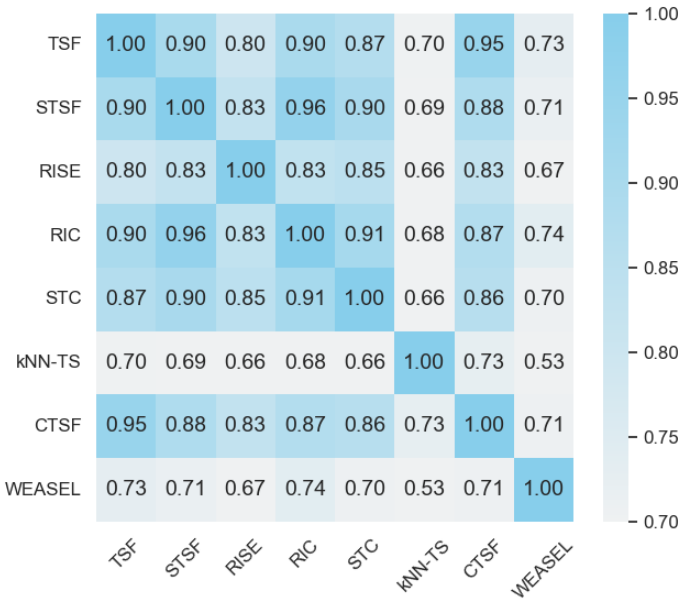


Fig.4.2A: Model correlation matrix. Similarity of the models in sport label predictions.

Model correlation

The model correlation matrix in figure 4.2A depicts similarity of classification model behavior. Number one (1) indicates identical model behavior, which means that models not only misclassify same segments in certain test data, but also has identical classification behavior. In other words, model correlation does not consider correct labels, but compares only prediction vectors. Number zero (0) indicates that models do not share a one single common misclassified segment. However, model correlation matrix does not tell anything about overall model performance in terms of accuracy but can be used to select optimal ones to create ensemble models, for example.

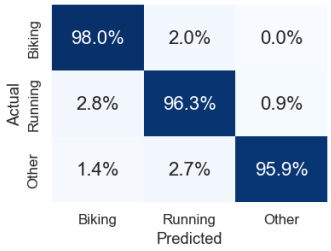
4.3 Multivariate TSC

M-TSC analysis introduction

Here we continue with the same frame as used for U-TSC model analysis using precision, recall and f1-score for each sport category distinctively and complemented by macro and weighted average considering categories altogether. However, we need to consider that the adopted M-TSC model deals with three 69 seconds signals, instead of one 207 second signal as in U-TSC classification. M-TSC and U-TSC use exactly the same data, which means the same information, but data structure

Table 4.3A: Multivariate Symbolic Extension (MUSE) classification report.

	Biking	Running	Other	Macro Avg	Weighted Avg
Precision	0.918	0.972	0.986	0.961	0.966
Recall	0.923	0.963	0.960	0.968	0.966
F1-Score	0.980	0.968	0.973	0.964	0.966
Support	49	109	74	232	232
Accuracy	**0.966**				



differs, and univariate models are not able to benefit from all the same information available, such as dimension correlation.

The MUSE proved to be a very effective model allowing up to 96,6% accuracy. From the classification report table 4.3A we can conclude that especially recognition of *running* and *other* categories was a great success and produced over 96% result for all the metrics precision, recall and f1-score. Biking seems to be the most problematic category for MUSE since precision is relatively lower with ~92% score, which obviously means these 8% of segments has been confused with *running* and *other*.

Ensemble Models

For CE model selection we implemented univariate classification for each of three features separately. Because the built CE model conducts independent feature classification, but ignores correlation information between them, the expected classification accuracy is apparently lower than MUSE's one. Table 4.3B shows the results of classification which was implemented using the same model setup as for univariate TSC in section 4.2, but instead of interlacing, using 69 seconds pure feature signals. In practice, the multivariate dataset dimensions *heart rate*, *speed*, and *altitude* were given separately as an input for the compiled classification algorithm A-3.4A, conducting total of three function calls. As the classification computation time of the CE fully depends on selected algorithms, we compromised and slightly weighted model *training-prediction* time over accuracy and ROC-AUC score. RIC algorithm was the most successful model for all the features in terms of accuracy and ROC-AUC, but by having tenfold time complexity in contrast to second best, STSF, and thus we decided to use RIC only for *altitude* feature where the accuracy gap was the largest. STSF was very close and within the variance of used evaluation metrics, and thus having only a marginal significance compared to a gained advantage in model training time, which was important factor with a limited study environment resources. Therefore, it was a consistent choice for two other features. Table 4.3C summarizes selected CE model setup STSF-STSF-RIC.

Table 4.3B: Classification results for each feature distinctively.

	HearRate		Speed		Altitude		
Classifier	**accuracy**	**roc-auc**	**accuracy**	**roc-auc**	**accuracy**	**roc-auc**	**Train/Test time ~ (s)**
STSF	0.582	0.736	0.922	0.986	0.750	0.913	**60 / 3**
RIC	0.582	**0.743**	**0.940**	**0.990**	**0.793**	**0.919**	**600 / 30**
STC	0.526	0.674	0.862	0.939	0.703	0.872	**900 / 30**
TSF	**0.591**	0.690	0.927	0.983	0.716	0.851	**10 / 1**
CTSF	0.543	0.676	0.914	0.981	0.664	0.844	**600 / 70**
RISE	0.513	0.671	0.858	0.973	0.707	0.889	**190 / 30**
WEASEL	0.509	0.653	0.909	0.974	0.711	0.844	**50 / 5**
kNN-TS	0.461	0.568	0.875	0.897	0.543	0.636	**2 / 12**
Selection	**STSF**		**STSF**		**RIC**		

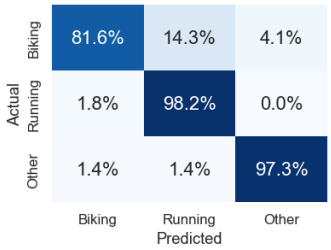
Table 4.3C: Column ensemble model classifiers. Best performing classifiers by features, by weighting time complexity.

Feature	**Selected Classifier**	**Accuracy**	**ROC-AUC**	**Training time (s) ~**
Heart Rate	Supervised Time Series Forest (STSF)	0.58	0.74	60
Speed	Supervised Time Series Forest (STSF)	0.92	0.99	60
Altitude	Random Interval Classifier (RIC)	0.79	0.92	600

As expected, CE placed between U-TSC and M-TSC models in classification performance by scoring 94,4% accuracy in a single algorithm execution. Many *biking* segments were misclassified, 14,3% as *running* and 4.1% as *other*, which in itself is fairly interesting because biking actually differs significantly from running based only on a mean value of speed feature.

Table 4.3D: Classification report of Column Ensemble (CE).

	Biking	**Running**	**Other**	**Macro Avg**	**Weighted Avg**
Precision	0.930	0.930	0.973	0.945	0.944
Recall	0.816	0.982	0.973	0.924	0.944
F1-Score	0.870	0.955	0.973	0.933	0.943
Support	49	109	74	232	232
Accuracy	**0.944**				



Hamming distance

There are several ways to compare performance between models using actual prediction instances such as the previously presented correlation matrix in figure 4.2A, which is probably the most traditional, but also hamming matrix and Cramer's V association. Next, we introduce hamming distance measure to support our model similarity analysis. Also, we include results from univariate model analysis in order to reflect results in the context of U-TSC part of the study.

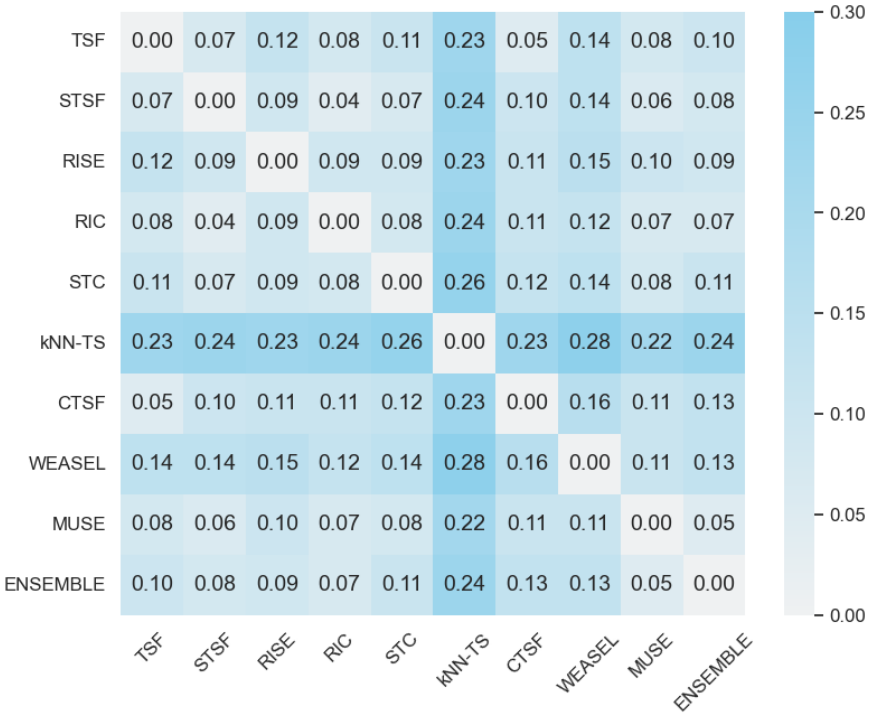


Fig. 4.3A: Hamming distance matrix. Dissimilarity in the classification between all sports.

For example, if accuracy of MUSE is 96.6% and accuracy of STSF is 93.1% it implies that the possible hamming distance range between models is:

$$\text{Max } (100 - 93.1) + (100 - 96.6) = 6.9 + 3.4 = 10.3\%$$

$$\text{Min } (100 - 93.1) - (100 - 96.6) = 6.9 - 3.4 = 3.5\%$$

The hamming distance between 1-D arrays u and v, is simply the proportion of disagreeing components in u and v. If u and v are boolean vectors, the hamming distance is:

$$\frac{C_{01} + C_{10}}{n} = \frac{mismatches}{number\ of\ all\ instances}$$

And since the test data size is 232, there can be disagreeing components at maximum $0.103 \cdot 232 \approx 24$ and at minimum $0.035 \cdot 232 \approx 7$. As a summary, hamming distance must be $[0.035 - 0.103]$. The produced distance value 0.06 from the figure 4.3A seems to be as expected. Thus, considering the example case with MUSE and STSF with 6% distance, models are behaving quite similarly having only 2.5% marginal to a maximum similarity with 3.5% minimum distance. In more general sense, it seems like WEASEL is performing well in the area where other models are not, because as an accurate model (91,8%) it has a big hamming distance value between other models with a similar accuracy. Models with the smallest distance value are STSF and RIC (4%). And if we ignore the exceptional

model kNN-TS, the longest distance is between WEASEL and CTSF by a value 16%, having only 2.1% accuracy difference.

DISCUSSION

In this section, we move one step back to the practical level and reflect gained results into the wider picture of the classification problem among sport activities considering also philosophical point of view.

5.1 Standard CML

Main part of the CML algorithms provide satisfactory results for SAC task among five particular activities. In certain random subsets of train and test data correct classification was achieved for all the instances by several models. As it was noticed in preliminary analysis, two main challenges were acknowledged: classification between walk and running activities might be challenging and even original data may have some questionable walking activities in which activity recorded as walking consist only 50-70% of walking and 30-50% of running segments. However, this would not be a problem if all the running activities were pure but in fact many running activities also have a significant amount of walking segments. That fact causes quite a deep intersecting area between walking and running activities. Another similar type of classification problem occurs between *R-Skiing* and *Skiing*. These activities are remarkably similar, but still they often need to be separated among active athletes. However, quite surprisingly these activities are well classified in general. But because a clear decision boundary was not found, there is yet room for discussion. Furthermore, LR and SVC models obtained a perfect decision boundary and classified them correctly in the first phase of model selection. And further, almost all other algorithms produced only one mistake – either *R-Skiing* classified as *Skiing* or controversary – and only two models misclassifying between these activities before parameter optimization. Biking activities were clearly separable by all the models, even with default hyperparameters, as we expected knowing its different characteristics.

When we investigate confusion matrices A-4.1A and A-4.1C with the same certain training and test data division using random state value 24, we can see that hyperparameters tuning fixed most of the problems when creating a decision boundary for *R-Skiing* and *Skiing*. For example, kNN, RF, QDA

were also able to classify these activities without confusion, among MLP and LR which worked perfectly even without hyperparameter optimization. Whereas LDA was the only model capable of separating *walking* and *running* without mistakes with used particular test data.

As a summary, all the included S-CML models produced good results having accuracy greater than 90%. In the introduction of this thesis, we discussed comprehensively results of the previous studies in HAR applications. There exist very few corresponding studies in a similar dataset used in this thesis due to a highly sensitive nature and therefore availability of data, results will be reflected in very general sense. Demrozi and others (2020) have made probably the largest study in recent years investigating the success and number of conducted HAR studies in the years 2015-2020 using both deep learning and classic machine learning implementations. A minority of the previous studies have used data produced by smartwatches. Among 96 CML studies average accuracies have been 92.2 percent which is the same as the weakest performing model in this study. However, the average number of activities has been 17, but in this study only 5, which logically makes classification task easier producing higher accuracies. [1] Also, many of the studies have included similar models such as kNN, SVC, RF, MLP or corresponding neural network, DT, LDA, QDA , and NB. Therefore, the results obtained in this thesis complement previous studies, especially when inertial sensor data is not available due to absence of sensors.

Lastly, it is reasonable to discuss the rationality of retrospective SAC using adopted method based on the results. Labelled data were used to define different activities. But strictly thinking, could we just conclude that misclassified cases of machine learning models are actually labeled incorrectly by human instead of interpreting them as a classification mistake. Let us take a hypothetical approach: *there exists a low intensity walk-like activity labeled as running which could be defined as a walking activity as well*. Therefore, if the goal is to make an algorithm to define sport activity type, it could not be even necessary to get results close to 100%. These decision boundaries could be set by algorithms, and then if an athlete goes for the intended running activity and labels it as a running type, algorithm could change that activity to be walking if the person in fact does not run more than certain predefined threshold value. Then, the case is not only to find a perfect model to separate activities but based on the boundaries which are set by machine learning model – learnt from personal user sport activity data – to help user to classify these activities or prevent intentional or unintentional mislabeling in social media segment leader boards, for instance.

Table 5.3A: Classification result compilation table. Models ordered by best accuracy score. Last column RTC means Relative Time Complexity in the value range [0=fast,10=slow]. For example, number 10 indicates tenfold train/fit time to the corresponding indicator value 1. Value zero indicates that there is no statistical significance in model computing time. Score variance is simple (max-min)/2.

Classifier	Type	ROC-AUC	Best Score	Score Variance	Mean Score	RTC [0,10]
MUSE	M-TSC	0.994	**0.966**	0.004	0.961	3
ENSEMBLE	M-TSC	0.995	**0.961**	0.002	0.958	8
STSF	U-TSC	0.990	**0.931**	0.004	0.925	1
RIC	U-TSC	0.985	**0.927**	0.006	0.920	6
WEASEL	U-TSC	0.981	**0.918**	0.000	0.918	1
GB	S-CML	-	**0.918**	0.004	0.918	0
MLP	NN	-	**0.918**	0.000	0.912	0
RISE	U-TSC	0.980	**0.905**	0.002	0.902	2
RF	S-CML	0.980	**0.905**	0.004	0.902	0
TSF	U-TSC	0.979	**0.901**	0.002	0.899	0.2
STC	U-TSC	0.973	**0.897**	0.004	0.892	10<
CTSF	U-TSC	0.978	**0.892**	0.002	0.889	10<
SVM	S-CML	-	**0.884**	0.000	0.884	0
DT	S-CML	0.898	**0.875**	0.013	0.871	0
LR	S-CML	-	**0.853**	0.000	0.853	0
kNN	S-CML	-	**0.849**	0.000	0.849	0
LDA	S-CML	-	**0.789**	0.000	0.789	0
G-NB	S-CML	-	**0.784**	0.000	0.784	0
kNN-TS	U-TCS	0.829	**0.772**	0.000	0.772	<1
QDA	S-CML	-	**0.573**	0.000	0.573	0

5.2 TSC

In TSC part we consider both univariate and multivariate classification results together. Based on the classification result on table 5.3A, we can conclude that TS classifiers performed successfully and produced enhanced accuracies in contrast to S-CML models, as expected. However, there are two striking exceptions or deviations: distance-based time series model *kNN-TS* with dynamic time warping does not work with constructed multivariate sequences in tolerable accuracy level. Using pure kNN which used Euclidean metric in distance computation was approximately 7.7% more accurate. On the controversary, two of the *sklearn*-models, GB and MLP produced better results than most of the univariate *sktime*-models. Considering GB and MLP model's relatively low time consumption for training, it obviously challenges *sktime*-models and their produced relevance gaining only 1-2% accuracy improvement in relation to their increased time complexity. Results show that MLP was also very stable and not sensitive to random changes in train and test data division. Further, according to

the results pure RF from *sklearn* scored higher accuracy than most of the *sktime*-models, especially TSF, although they use the same decision tree as a base model. This obviously seems to question the functionality of the adopted interlaced univariate signals, or advantage of using time series data in SAC.

Overall score level for U-TSC

Achieving over 90% accuracy score by several time series models in multivariate signals is close to the expected level or higher, as it was known that dataset has probably 5% (±2%) wrong or ambiguously labelled multi-sport activities. Analysis also included models which as such are not applicable to the constructed dataset format and require careful model hyperparameter optimization, and therefore have potential to improve their general performance in the dataset. To name a few, these are distance based kNN models, QDA, LDA, and G-NB, having accuracies from 54% to 85%.

Best models with interlaced univariate signals

Interval based model STSF (93,1%) and feature based model RIC (92,7%) were two best performing classifiers considering all used performance evaluation metrics, except time complexity. In relation to all other univariate models, STSF scored the best using *ROC-AUC,* and *mean accuracy* metrics, and outperforms the second most accurate model RIC also in terms of time, variance, and standard deviation of accuracies among iterations. RIC as a more time consuming and requirement of computational resources seems to make STSF preferable option when choosing a model for this type of classification task. In the study of Cabello et al. (2020), they proposed STSF as a supervised interval search technique with feature ranking metrics that is a highly efficient and accurate interval based TSC method. According to them, it minimizes unnecessary computations when finding relevant intervals, making it an order of magnitude faster than TSF. Further, by extracting relevant intervals from the trained trees, the generated regions of interest, i.e., groups of relevant intervals, highlight the time periods where differences in shape, distribution, or level, between time series of different classes are more pronounced, and thus provide interpretable outcomes of the classification. They stated that an extensive experimental study on 85 real datasets shows that STSF achieves classification accuracies competitive to state-of-the-art methods, but it is at least two orders of magnitude faster than them. This makes STSF suitable for large datasets with long time series. [29] The results of this study confirm their contributions in TSC in terms of accuracy, but not in computation time. Despite the statement STSF should minimize unnecessary computations, it was significantly slower in the dataset constructed. See the test environment here A-TE.

Misclassification and model correlation

Using the actual prediction data, we were able to conduct practical analysis to see which specific segments have been misclassified. In this study it was an especially useful tool to find highly likely by human misclassified activities. For example, if all of the models fail to classify some segment as in the table 4.2K three first instances, it has probably wrong label in original data. This study does not implement deeper analysis in this field, but it points out reasons which are probably decreasing or fluctuating accuracy. Also, misclassification helped to determine the essence of data.

By model correlation and distance analysis, similarities – as well as differences – between models were investigated. The best model of the study, STSF classified segments with 96% similarity with the second-best RIC but correlates up to 90% level with TSF, RISE, STC, and CTSF. Especially in correlation analysis was found, that WEASEL as a one of the best univariate models performs in an exceptional way and may thus produce great advantage when classifying those problematic misclassified activities when building ensemble models, for example. In general, this analysis may provide useful information for further classification applications in SAC studies, or when investigating model specific characteristics.

Multivariate classification

Adopted two multivariate classification algorithms are unquestionably the most accurate models in the study case dataset of a single athlete. The MUSE algorithm was not only the most accurate (96,6%), but was able to maintain stability among iterations, achieved the second highest ROC-AUC evaluator metric value, and at the same time, having low time complexity by our model-relative indicator value of 2. MUSE was a relatively fast model outperforming several univariate ones.

The second model for M-TSC was a custom ensemble model compiled from the best performing univariate algorithms per feature in separate individual feature classification. The implemented STSF-STSF-RIC column ensemble model performed better than expected in relation to MUSE being very close by 96,1% mean accuracy. Expectations for ensemble were based on the knowledge that it considers each feature as independent ignoring potentially important feature relationship information. Furthermore, ensemble classifier produced the best ROC-AUC score that was used as a main model quality indicator in TSC. Good performance, especially in relation to MUSE that considers feature correlation information, probably indicates that our three dataset features have low level dependency, as can be seen from the feature correlation matrix A-2.1C, or the dependency is similar among categories producing poor distinctiveness. The only disadvantage of the model was its time complexity,

indicator value 5 which comes from 1+1+3 for STSF-STSF-RIC combination of models. That requirement for computational resources is a typical disadvantage for ensemble models.

In this thesis we reinforced that multivariate methods are able to enhance classification accuracy in relation to univariate models without increasement in time complexity. Also, we noticed that ensemble models can achieve almost as good accuracy as adopted MUSE classifier, but increased time complexity is directly relative to the dimensionality of data and naturally depends on the selected ensemble models and their characteristics.

5.3 Summary

In this section, classification results are discussed including both conducted datasets in the study case. Since classification was implemented using exceptional datatype for HAR studies, there are very few comparable previous studies to reflect achieved results. Therefore, we rather avoid too straightforward comparisons being aware of the risks of misinterpretations. This unique retrospective method for personalized sport activity classification is probably not comparable at all, and in its extent, there is no actual need for it, apart from the aspects raised in the introduction part of this thesis. In general, countless previous studies – mentioned in introduction section – have achieved at least the critical 90 percent accuracy in HAR, and even up to 99 percent in the datasets created under “sterile” controlled laboratory conditions. Multivariate TSC accuracy of 96,6% by MUSE, and univariate TSC accuracy of 93,1% by STSF observed in this thesis is very satisfying and promising for future studies to develop a well performing model for a retrospective personalized outdoor sport activity classification using only three particular features.

As a summary, ten well known CML models were included in TSC analysis in order to reflect model performance and gained advantage of TSC compared to traditionally very fast feature based classification algorithms which ignores sequence value order. Data used in these models was in tabularized format of the same interlaced signals. The main difference in contrast to TSC models is that they consider each value of the segment as an independent feature value. Therefore, S-CML models provide the most important reference point in order to gain understanding about the real benefit of the usage of time series for SAC, in general.

In comparison to S-CML models, was found that Gradient Boosting, and Multilayer Perceptron produced very competitive results with even the best U-TSC models. Also, Random Forest, which was the base algorithm for most of the U-TSC models and as such produced very competitive results with 90,5% accuracy. However, although it seems like standard machine learning models are very

close to TSC models we must be careful in making generalized conclusions from the results. The real advantage of TSC must be tested using the data from several people with different characteristics considering interpersonal differences.

Performance map for TSC models

As adopted S-CML models do not have a statistically significant computational time in this particular small dataset case due to their feature based classification method, they can be ordered quite unambiguously using only accuracy values. Since TSC models are dealing with much more data it becomes extraordinarily important to take into account also the time complexity of the algorithms. Probably one of the best ways to visualize this is to use scatter diagram as in figure 5.3A. It describes model accuracies in relation to the combined training and testing time of the models. For example, STC and CTSF, which are located on the right, require much more time, and achieve accuracy less than average. KNN-TS is quite fast but performs very poorly in terms of accuracy. Based on the results, those models may not be preferable to use in future TS-SAC studies. Ensemble performs very well, but respectively, the time complexity is in a "grey" area. Further, the RIC model is competitive by accuracy but needs fivefold more time than STSF and WEASEL. Therefore, according to the results we may suggest MUSE, STSF, WEASEL, TSF, and RISE for future studies in TS-SAC using selected particular feature set, heart rate, speed, and altitude.

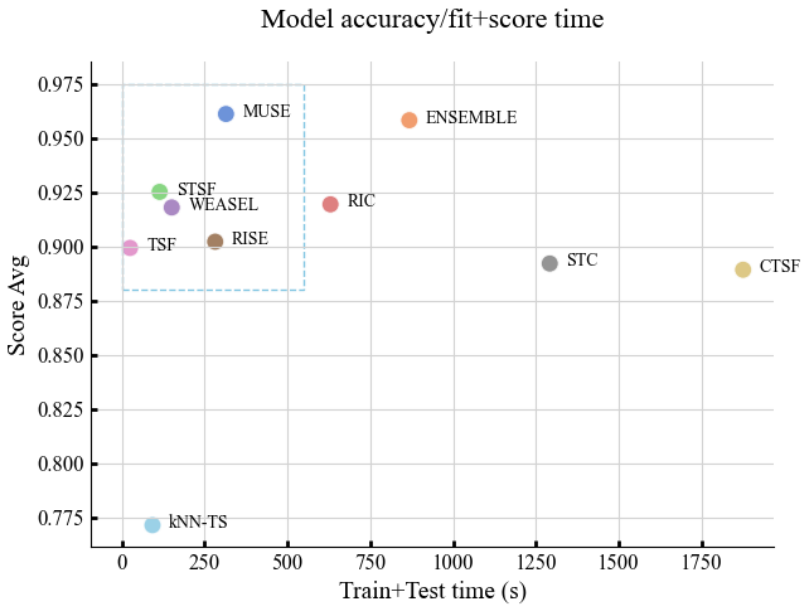


Fig. 5.3A: TSC model performance map. The upper left corner indicates perfect accuracy-time relation (fast and accurate model). The lower right corner indicates long execution time with low accuracy.

CONCLUSIONS

Three diverging methods were proposed and investigated to conduct a *retrospective personalized supervised sport activity classification* of a single person by training model with the sport history data of the athlete. Based on the literature overview we found that human activity recognition has been well studied and good results up to 99% achieved in recent years. However, in most of the cases studies have been conducted in datasets created in laboratory conditions, and such an environment hardly corresponds the reality. Whereas in this thesis we were able to implement machine learning application in the data generated by athlete who has not been conscious that data will be used for the study purposes. This environmental fact is the most significant difference, and therefore probably will create a specific value for this thesis work. In other hand, it is also the biggest weakness of the thesis, since results might not be generalizable, and they may differ remarkably among different persons. But still, this thesis shows, that in certain conditions it is possible to classify sport activities successfully using adopted machine learning algorithms and only certain pre-selected features, such as heart rate, speed, and altitude.

Two different datasets with diverging structure and type of the same sport activities were used, namely traditional and consistent small size summary dataset which can be stored unambiguously in a single comma separated value file, and much more complicated time series dataset, wherein data is stored in a separate files and it requires a great effort to transform it into format to be processed by computer and applied TSC algorithms. This thesis presented the whole process of structuring such fragmented data step by step in order to perform time series classification task.

A total of twenty different machine learning models from *sklearn* and *sktime* were adopted which makes the study quite extent application of CML models into a specific type of sport data. The observed results are mainly good, and even better than the average results in recent HAR studies (Demrozi et al., 2020)([1]). The performance of the lately developed multivariate algorithm MUSE, which exploits a word extraction from the signals of multivariate data technique, was acknowledged, and can be gently proposed as the best choice for sport activity classification using multivariate time series data. In univariate data there are two competitive models among WEASEL, which is the univariate version of the MUSE, namely STSF and RIC. However, the most important question, which is whether to use time series data or traditional data, was not answered based on the results in this thesis. As the same accuracy level was achieved using M-TSC and S-CML, we may set a question which is the second most important metric to measure performance? In the world of computers and

bits, the issue can be stated basically like this: how to achieve correct and reliable answers as quickly as possible. Therefore, we must start with the question, what kind of data is available? If both datasets are available, then we suggest using S-CML as a very fast and effective method with low computation requirement. Whereas in the case of time series data from the heart rate, speed, and altitude sensors, then the question will be more complex: whether to extract features to transform dataset into classical machine learning task or apply it as a time series classification problem using multivariate models. However, there seems to be insufficient evidence for suggestion to implement TS-SAC applying interlacing method for multivariate signal if the accuracy metric is critical and 2-3% magnitude difference has significance in the classification task. Still, in certain environments and circumstances, TSF as a fast model could be more preferable than MUSE, and then solving this problem in introduced interlaced univariate data can take place.

6.1 Standard CML

In the S-CML part of study, nine diverse types of standard CML algorithms were tested for sport dataset with 5 categories and 21 features. The lack of publicly available datasets in the sport activity field was acknowledged, and that is the main problem in order to perform more comprehensive analysis. However, there was available a small size single athlete dataset in high quality with correctly labelled instances. Also, the dataset was successfully transformed to the suitable form for CML model evaluation analysis. Model performances were evaluated in standard and PCA data format. t-SNE worked well as a data visualization tool where we obtained that sport activities are quite clearly separable. Dataset contained overlapping instances with deep intersect area and unexpected single data points which could be considered as misclassified cases by human, or they have otherwise exceptional data features. Those single cases caused the main accuracy decrease, and high accuracy variance in the models when splitting data randomly into train and test splits. In the used particular dataset, no benefit was achieved either by using principal components. The average situation was controversary, and all the models performed either equal or better when using only standardized dataset features.

Several well performing models were found, and in general mean accuracy among 20 classification iterations was within quite a narrow range between 94.6% - 96.6%. With specific data division sets almost all the models achieved 100% classification accuracy which partly indicates anomalies in the data. In addition to the small data size, the accuracy measure was sensitive to anomalies causing high variability in the results.

6.2 TSC

In TSC part of the study, including U-TSC and M-TSC, one of the most significant issues was dataset construction, or transformation from the separate files to *univariate data* to be classified with univariate time series classification algorithms. In order to create univariate data from three-dimensional sequences, *multivariate data* was transformed into *interlaced univariate signals.* That method enabled us to perform SAC using univariate TSC models. Data was successfully classified, achieving a relatively high overall accuracy level 90±3% among most of the used TSC algorithms. Considering dataset quality, and likely classification errors in known data, we found that it is possible to achieve accuracy of up to 97-99% in the classification, taking into account the results of the misclassification analysis.

Dataset consisted of up to 5 sport categories from which *skiing*, *roller skiing* and *walking* were combined in the *other* category. Thus, we compressed categories a little bit from five to three in order to simplify classification task. This was also done partly for technical reasons. Since in this study we emphasized dataset construction procedure and analyzed how successfully it can be applied in the study case, the main focus was on developing a well functioning method for TSA-SAC. Therefore, the nature of the study was preliminary, and it leaves a lot of room for expansions and to further develop the implemented method. However, good results were achieved, and this established a good foundation for the application of TSC algorithms to a larger and more diverse dataset in future studies.

6.3 Socio-philosophical perspective

From a socio-philosophical perspective, after this analysis we may propose that SAC itself is not problematic and decision boundaries of well performing models could be used with good result expectation in retrospective SAC analysis tasks for sport activity recording devices and their ecosystems. In social media platforms for athletes, human labelled sport activities could be corrected if adopted machine learning model will observe or consider it to be closer to another activity. Using isolated applications for sport activity classification in social sport activity platforms could be a great solution without prejudice to the right of an individual athlete to classify, for example, walking-like activities as running activities. In sport activities for social communities, the choice made by a machine to determine its type may be more acceptable because it treats the parties equally. That fact may also reconcile rising ethical-moral issues as decisions made by machines will be in a higher priority than human made decisions.

6.4 Future work

For future investigations in SAC, we suggest an unsupervised approach to explore how well SAC will function as a decision boundary method for retrospective category correction problems. Further, there is likely great potential in deep learning and neural network algorithms when the size of the dataset increases. As multivariate models performed with the best accuracy in our dataset, it will be the most interesting field to contribute more comprehensive investigations with model and dataset optimization, paying more attention to where misclassifications were made, and especially which kinds of actual instances could be problematic when classifying sports. Yet, one interesting field for SAC is to conduct classification using data from several athletes and to investigate what the effect is when the same activities are performed by several different athletes.

REFERENCES

1. Demrozi F. et al. Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey // IEEE Access. 2020. Vol. 8. P. 210816–210836.
2. Hsu Y.-L., Chang H.-C., Chiu Y.-J. Wearable Sport Activity Classification Based on Deep Convolutional Neural Network // IEEE Access. 2019. Vol. 7. P. 170199–170212.
3. Mekruksavanich S., Jitpattanakul A. Smartwatch-based Human Activity Recognition Using Hybrid LSTM Network // 2020 IEEE SENSORS. Rotterdam, Netherlands: IEEE, 2020. P. 1–4.
4. Lara O.D., Labrador M.A. A Survey on Human Activity Recognition using Wearable Sensors // IEEE Commun. Surv. Tutor. 2013. Vol. 15, № 3. P. 1192–1209.
5. Tapia E.M. et al. Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor // 2007 11th IEEE International Symposium on Wearable Computers. Boston, MA, USA: IEEE, 2007. P. 1–4.
6. Foerster F., Smeja M., Fahrenberg J. Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring // Comput. Hum. Behav. 1999. Vol. 15, № 5. P. 571–583.
7. Berchtold M. et al. An Extensible Modular Recognition Concept That Makes Activity Recognition Practical // KI 2010: Advances in Artificial Intelligence / ed. Dillmann R. et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Vol. 6359. P. 400–409.
8. Lara Ó.D. et al. Centinela: A human activity recognition system based on acceleration and vital sign data // Pervasive Mob. Comput. 2012. Vol. 8, № 5. P. 717–729.
9. Ahmed N., Rafiq J.I., Islam M.R. Enhanced Human Activity Recognition Based on Smartphone Sensor Data Using Hybrid Feature Selection Model // Sensors. 2020. Vol. 20, № 1. P. 317.
10. Micucci D., Mobilio M., Napoletano P. UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones // Appl. Sci. 2017. Vol. 7, № 10. P. 1101.
11. Mitchell E., Monaghan D., O'Connor N. Classification of Sporting Activities Using Smartphone Accelerometers // Sensors. 2013. Vol. 13, № 4. P. 5317–5337.

12. Wang Y. et al. Volleyball Skill Assessment Using a Single Wearable Micro Inertial Measurement Unit at Wrist // IEEE Access. 2018. Vol. 6. P. 13758–13765.
13. Hsu Y.-L. et al. Human Daily and Sport Activity Recognition Using a Wearable Inertial Sensor Network // IEEE Access. 2018. Vol. 6. P. 31715–31728.
14. Xie L. et al. Human activity recognition method based on inertial sensor and barometer // 2018 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL). Moltrasio: IEEE, 2018. P. 1–4.
15. Ghazali N.F. et al. Common sport activity recognition using inertial sensor // 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA). Batu Feringghi: IEEE, 2018. P. 67–71.
16. Fang L., Yishui S., Wei C. Up and down buses activity recognition using smartphone accelerometer // 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. Chongqing, China: IEEE, 2016. P. 761–765.
17. Yin X. et al. Human activity detection based on multiple smart phone sensors and machine learning algorithms // 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD). Calabria, Italy: IEEE, 2015. P. 582–587.
18. Rassem A., El-Beltagy M., Saleh M. Cross-Country Skiing Gears Classification using Deep Learning. arXiv, 2017.
19. Espinilla M. et al. Human Activity Recognition from the Acceleration Data of a Wearable Device. Which Features Are More Relevant by Activities? // UCAmI 2018. MDPI, 2018. P. 1242.
20. Rokni S.A., Nourollahi M., Ghasemzadeh H. Personalized Human Activity Recognition Using Convolutional Neural Networks // Proc. AAAI Conf. Artif. Intell. 2018. Vol. 32, № 1.
21. Zhuang Z., Xue Y. Sport-Related Human Activity Detection and Recognition Using a Smartwatch // Sensors. 2019. Vol. 19, № 22. P. 5001.
22. Faouzi J. Time Series Classification: A review of Algorithms and Implementations, Classification de séries temporelles : une revue d'algorithmes et d'implémentations // Machine Learning (Emerging Trends and Applications) / ed. Kotecha K. Proud Pen, 2022.
23. Pareek J., Jacob J. Data Compression and Visualization Using PCA and T-SNE // Advances in Information Communication Technology and Computing / ed. Goar V. et al. Singapore: Springer Singapore, 2021. Vol. 135. P. 327–337.
24. Raschka S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. arXiv, 2018.
25. Löning M. et al. Sktime: A Unified Interface for Machine Learning with Time Series. arXiv, 2019.
26. Bagnall A. et al. A tale of two toolkits, report the first: benchmarking time series classification algorithms for correctness and efficiency. arXiv, 2019.
27. Chaudhary A., Kolhe S., Kamal R. An improved random forest classifier for multi-class classification // Inf. Process. Agric. 2016. Vol. 3, № 4. P. 215–222.
28. Deng H. et al. A time series forest for classification and feature extraction // Inf. Sci. 2013. Vol. 239. P. 142–153.
29. Cabello N. et al. Fast and Accurate Time Series Classification Through Supervised Interval Search // 2020 IEEE International Conference on Data Mining (ICDM). Sorrento, Italy: IEEE, 2020. P. 948–953.

30. Flynn M., Large J., Bagnall T. The Contract Random Interval Spectral Ensemble (c-RISE): The Effect of Contracting a Classifier on Accuracy // Hybrid Artificial Intelligent Systems / ed. Pérez García H. et al. Cham: Springer International Publishing, 2019. Vol. 11734. P. 381–392.
31. Bagnall A. et al. On the Usage and Performance of the Hierarchical Vote Collective of Transformation-Based Ensembles Version 1.0 (HIVE-COTE v1.0) // Advanced Analytics and Learning on Temporal Data / ed. Lemaire V. et al. Cham: Springer International Publishing, 2020. Vol. 12588. P. 3–18.
32. Rodriguez J.J., Kuncheva L.I., Alonso C.J. Rotation Forest: A New Classifier Ensemble Method // IEEE Trans. Pattern Anal. Mach. Intell. 2006. Vol. 28, № 10. P. 1619–1630.
33. Schäfer P., Leser U. Fast and Accurate Time Series Classification with WEASEL // Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. Singapore Singapore: ACM, 2017. P. 637–646.
34. Bagnall A. et al. A tale of two toolkits, report the third: on the usage and performance of HIVE-COTE v1.0. arXiv, 2020.
35. Ruiz A.P. et al. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances // Data Min. Knowl. Discov. 2021. Vol. 35, № 2. P. 401–449.
36. Schäfer P., Leser U. Multivariate Time Series Classification with WEASEL+MUSE. arXiv, 2017.

APPENDICES

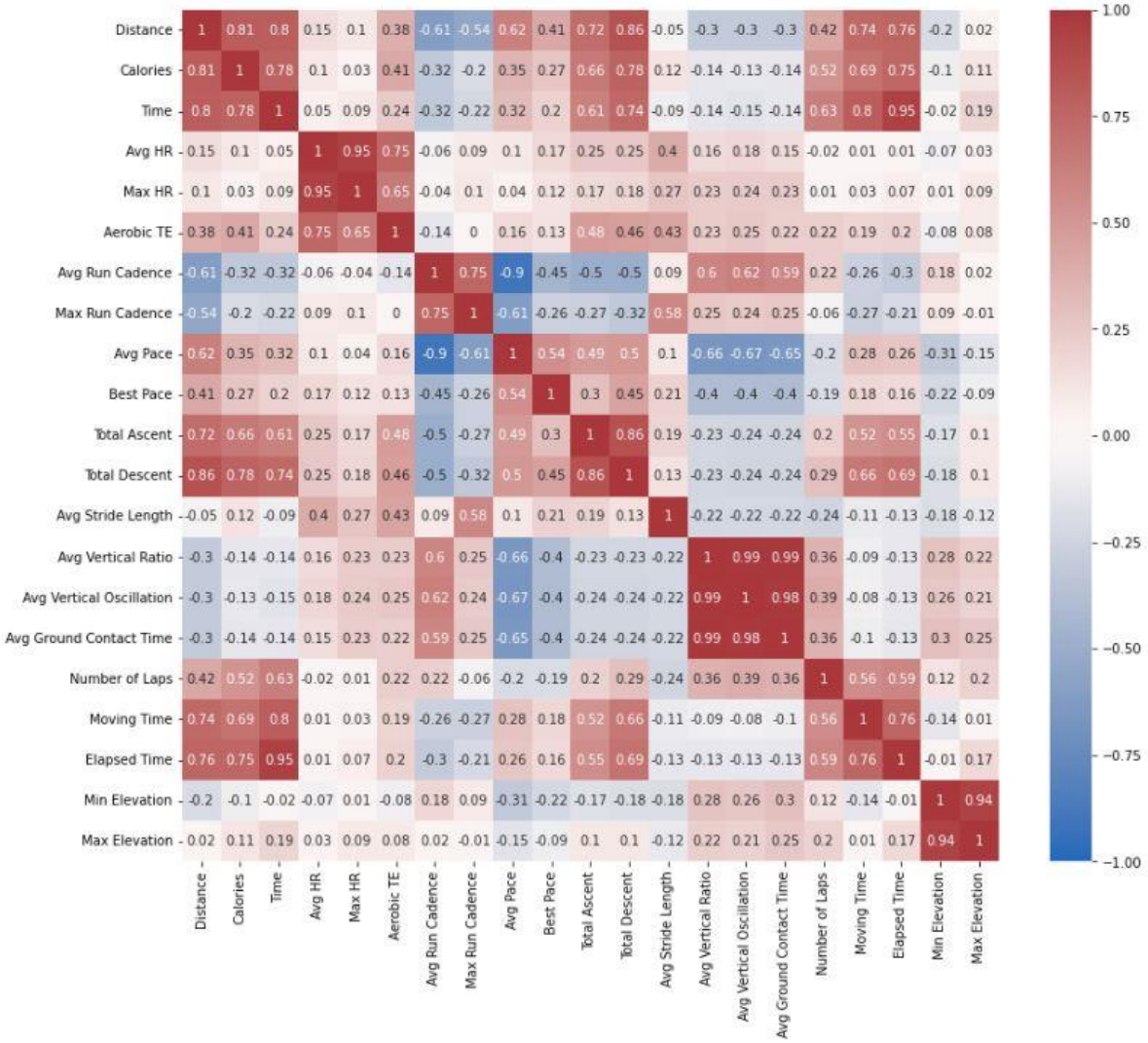
A-2.1A: ORIGINAL DATASET

```
RangeIndex: 220 entries, 0 to 219
Data columns (total 40 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Activity Type              220 non-null    object
 1   Date                       220 non-null    object
 2   Favorite                   220 non-null    bool
 3   Title                      220 non-null    object
 4   Distance                   220 non-null    float64
 5   Calories                   220 non-null    object
 6   Time                       220 non-null    object
 7   Avg HR                     220 non-null    object
 8   Max HR                     220 non-null    object
 9   Aerobic TE                 220 non-null    object
 10  Avg Run Cadence            220 non-null    object
 11  Max Run Cadence            220 non-null    object
 12  Avg Pace                   220 non-null    object
 13  Best Pace                  220 non-null    object
 14  Total Ascent               220 non-null    object
 15  Total Descent              220 non-null    object
 16  Avg Stride Length          220 non-null    float64
 17  Avg Vertical Ratio         220 non-null    float64
 18  Avg Vertical Oscillation   220 non-null    float64
 19  Avg Ground Contact Time    220 non-null    object
 20  Avg GCT Balance            220 non-null    object
 21  Training Stress Score®     220 non-null    float64
 22  Grit                       220 non-null    float64
 23  Flow                       220 non-null    float64
 24  Total Strokes              220 non-null    object
 25  Avg. Swolf                 220 non-null    object
 26  Avg Stroke Rate            220 non-null    object
 27  Total Reps                 220 non-null    object
 28  Total Sets                 220 non-null    object
 29  Dive Time                  220 non-null    object
 30  Min Temp                   220 non-null    float64
 31  Surface Interval           220 non-null    object
 32  Decompression              220 non-null    object
 33  Best Lap Time              220 non-null    object
 34  Number of Laps             220 non-null    int64
 35  Max Temp                   220 non-null    float64
 36  Moving Time                220 non-null    object
 37  Elapsed Time               220 non-null    object
 38  Min Elevation              220 non-null    object
 39  Max Elevation              220 non-null    object
dtypes: bool(1), float64(9), int64(1), object(29)
memory usage: 67.4+ KB
```

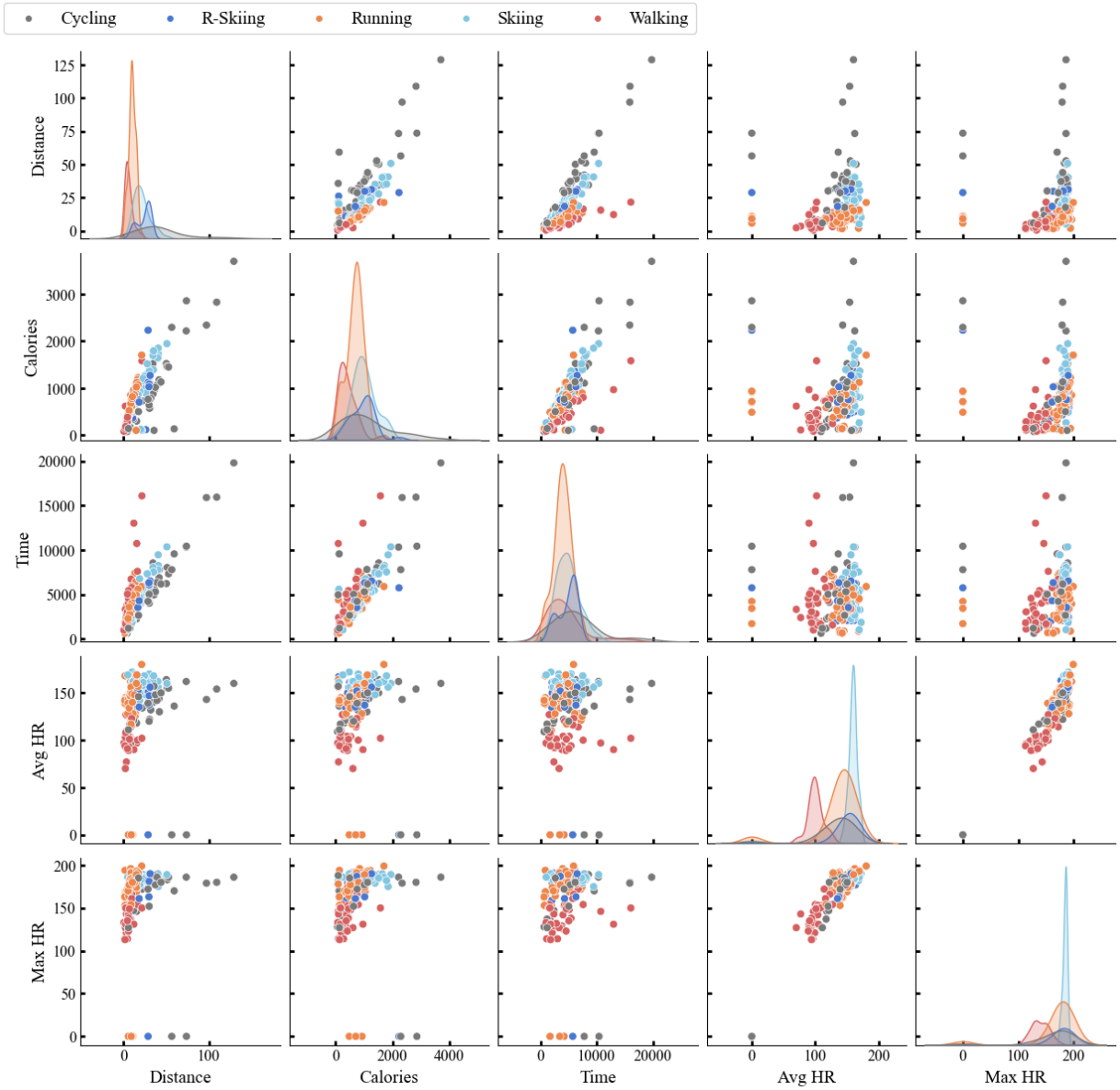
A-2.1B: CLASSIFICATION DATASET

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213 entries, 0 to 212
Data columns (total 22 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Activity Type              213 non-null    object
 1   Distance                   213 non-null    float64
 2   Calories                   213 non-null    int32
 3   Time                       213 non-null    float64
 4   Avg HR                     213 non-null    float64
 5   Max HR                     213 non-null    float64
 6   Aerobic TE                 213 non-null    float64
 7   Avg Run Cadence            213 non-null    float64
 8   Max Run Cadence            213 non-null    float64
 9   Avg Pace                   213 non-null    float64
 10  Best Pace                  213 non-null    float64
 11  Total Ascent               213 non-null    float64
 12  Total Descent              213 non-null    float64
 13  Avg Stride Length          213 non-null    float64
 14  Avg Vertical Ratio         213 non-null    float64
 15  Avg Vertical Oscillation   213 non-null    float64
 16  Avg Ground Contact Time    213 non-null    float64
 17  Number of Laps             213 non-null    int64
 18  Moving Time                213 non-null    float64
 19  Elapsed Time               213 non-null    float64
 20  Min Elevation              213 non-null    float64
 21  Max Elevation              213 non-null    float64
dtypes: float64(19), int32(1), int64(1), object(1)
memory usage: 35.9+ KB
```

A-2.1C: FEATURE CORRELATION MATRIX

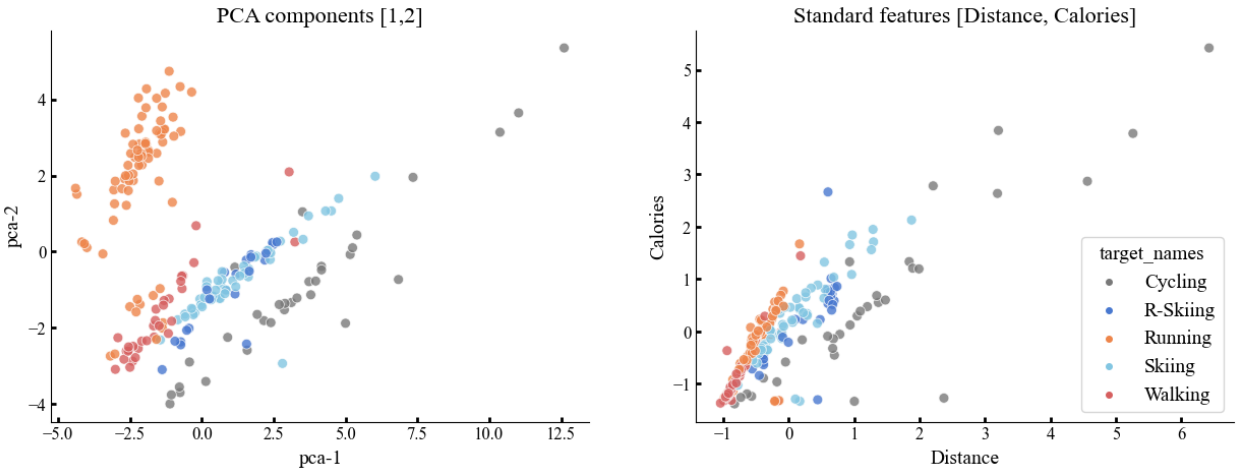


A-2.1D: ORIGINAL FEATURE VALUE PAIR PLOTS



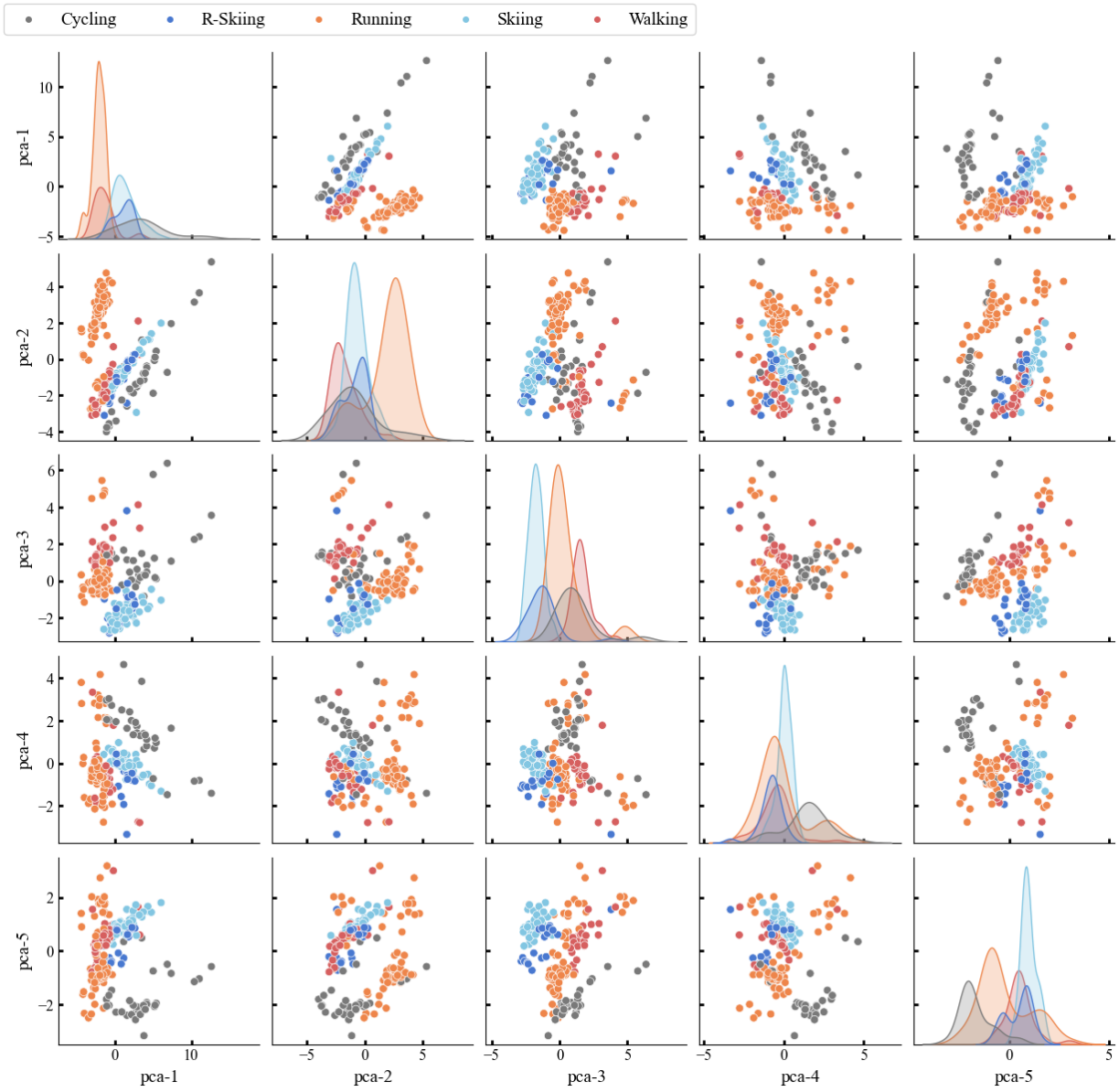
Original feature value pair plots (5 first features)

A-2.1E: PCA AND STD FEATURE CORRELATION PLOTS



Correlation of two first features in PCA and standard data

A-2.1F: PCA PAIR PLOTS



PCA pair plots among five most principal components

A-2.2A: TCX TO CSV TRANSFORMER

Python source code: TCX to CSV file transformer

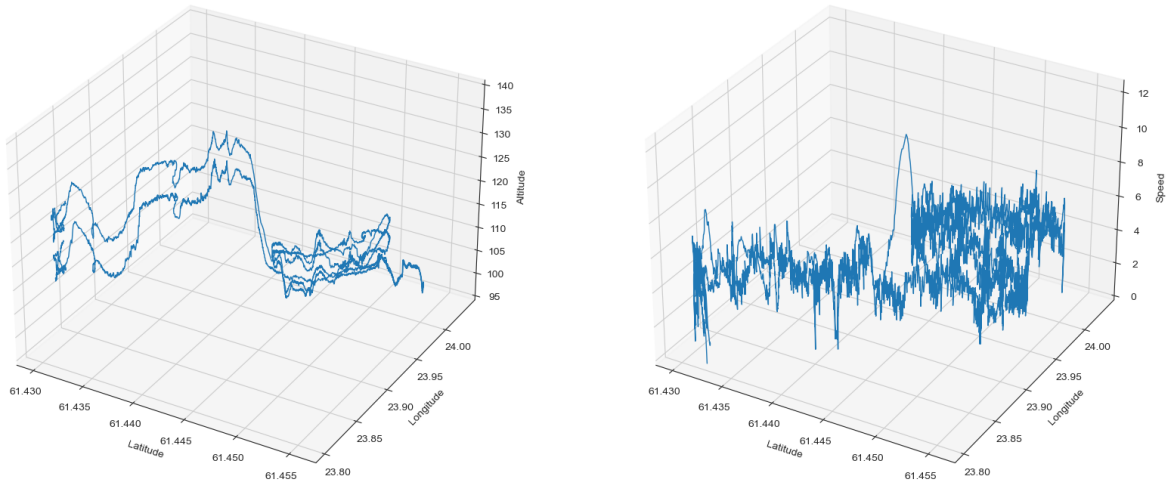
```
# tcxreader instructions: https://pypi.org/project/tcxreader/
Import datetime
import numpy as np
from tcxread import TCXReader, TCXTrackPoint, TCXExercise
from filelocations import Dir
from pathlib import Path
# from tabulate import tabulate # For data visualization in console

def tcx_trackpoints_to_numpy(file_path: str, column_types: [(str,type)]) -> tuple:
    """
    TCX Reader - Reading a single .tcx file to NumPy array
    @param file_path: File path for tcx file to read
    @param column_types: Column/Variable types and/or names. Will be directly stored as a
    value of numpy array function 'dtype' parameter
    @return: Two dimensional NumPy array containing all the rows and columns from 'file_path'
    """
    tcx_reader = TCXReader()
    data: TCXExercise = tcx_reader.read(file_path)
    sport = data.activity_type
    print('Getting trackpoints for activity type: %s' % sport)
    trackpoints = data.trackpoints
    print('The file has ' + str(len(trackpoints)) + ' trackpoints')

    # Creates empty numpy array with 9 columns
    tp_array = np.empty((0, len(column_types)), dtype=column_types)
    # Constructing NumPy array
    for tp in trackpoints:
        datarow = [tp.time, tp.latitude, tp.longitude, tp.elevation,
                   tp.distance, tp.hr_value, tp.TPX_speed, tp.cadence, tp.watts]
        tp_array = np.append(tp_array, np.array([datarow]), axis=0)

    return (sport, tp_array)
```

```
def tcx_exercises_to_csv(tcx_folder: str, csv_folder: str,
                         column_types: [(str,type)], column_names: str,
                         exers: int) -> list:
    """
    TCX Reader - Reads a set of .tcx files from the given path and writes them to
    file using function save_tcx_to_csv()
    @param column_types: Data variable types (float, int, datetime, str).
    @param column_names: Data variable names for .csv header row.
    @param exers: Number of activities/exercises to read from .tcx file set
    @return: Activity types as list of strings in a size of given function parameter
    "exers" for classification tasks
    """
    tcx_files = Path(tcx_folder).glob('*.tcx')
    count = 0

    for child in tcx_files:
        count+=1
    tcx_files = Path(tcx_folder).glob('*.tcx')
    activity_types = np.array([])
    file_number = 1

    # HERE WE HANDLE TOO BIG NUMBER OF EXERCISES VALUE
    if exers == 0 or exers > count:
        print('exers 1: ' + str(exers))
        exers = count
        print('exers 2: ' + str(exers))

    for child in tcx_files:  # In order to avoid reading errors
        if child.is_file() and file_number <= exers:
            print('\nFile number to process: ' + str(file_number))
            print('Activity inserted: ' + child.name
                + ', Activities = ' + str(file_number))
            # Data instance
            r_path_filename = tcx_folder + child.name
            w_path_filename = csv_folder + child.name
            sport = save_tcx_to_csv(r_path_filename, w_path_filename,
                                    column_types, column_names)
            activity_types = np.append(activity_types, sport)
            file_number = file_number + 1
        else:
            break

    return activity_types
```

```
def save_tcx_to_csv(read: str, write: str, types: [(str,type)], names: str) -> str:
    """
    Converts garmin .tcx file to .csv.
    @param read: Tcx file path to read from
    @param write: Csv file path to write to
    @param types: Data variable types (float, int, datetime, str).
    @param names: Data variable names for .csv header row. Values separated by semicolon.
    @return: String/Name of the saved sport activity
    """
    sport, tp_array = tcx_trackpoints_to_numpy(read, types)
    write = str(Path(write).with_suffix('.csv'))
    print('Read from: ' + read + '\nWrite to: ' + write)
    np.savetxt(write, tp_array, delimiter=';', fmt='% s', header=names, comments='')

    # Let's return activity type since we won't handle it here
    return sport


def save_tcxset_to_csv(root: str, read: str, write: str, types: [(str,type)], names: str,
                       exers: int=0) -> object:
    """
    Converts set of garmin .tcx files to one single .csv file.
    @param root: Root folder for all the data (read and write)
    @param read: Tcx folder path to read
    @param write: Csv file path/name to write
    @param types: Data variable types (float, int, datetime, str).
    @param names: Data variable names for .csv header row. Values separated by semicolon.
    @param exers: Number of activities to read from .tcx set
    """
    activity_types = tcx_exercises_to_csv(root + read, root + write, types, names, exers)
    print(activity_types)
    write = str(Path(root + write + 'activity_types.csv'))
    print('\nSaving activity types to: ' + write)
    np.savetxt(write, activity_types, delimiter=';', fmt='% s',
               header='activity_name', comments='')
```

```
from filelocations import Dir
from datetime import datetime
from activityparser import save_tcx_to_csv, save_tcxset_to_csv
import sys

def main():
    args = sys.argv
    args = args[1:]  # First element of args is the python.py file name to run
    print(args)

    if len(args) < 1:
        print('Exactly two command line arguments or "--help" are required!')
        print('--Also, only two first arguments are noticed, others are ignored')
    elif len(args) == 1:
        if args[0] == '--help':
            printInfo()
        else:
            print('Unknown command!')
    elif len(args) >= 2:
        c_type = args[0]
        if c_type == 's':
            # Saving single TCX activity to CSV file
            filename = args[1]
            time_stamp = datetime.now()
            save_tcx_to_csv('.' + Dir.TCX_PATH.value + filename,
                            Dir.CSV_PATH.value + filename,
                            Dir.COLUMN_TYPES_SINGLE.value, Dir.NAMES_SINGLE.value)
            print('Single TCXExercise processing time: ' + str(datetime.now() - time_stamp))
        elif c_type == 'm':
            activity_number = int(args[1])
            print(activity_number)
            # Saving set of TCX activities to CSV file
            root_path = args[2]
            time_stamp = datetime.now()
            # File paths
            print(root_path)
            print(Dir.TCX_PATH_SET1.value)
            print(Dir.CSV_PATH_SET1.value)
            save_tcxset_to_csv(root_path, Dir.TCX_PATH_SET1.value,
                               Dir.CSV_PATH_SET1.value, Dir.COLUMN_TYPES_SET.value,
                               Dir.NAMES_SET.value, activity_number)
            print('\nSet of TCXExercises processing time: '
                  + str(datetime.now() - time_stamp))
```

A-2.2B: 3D VISUALIZATION OF SPORT ACTIVITY SAMPLE



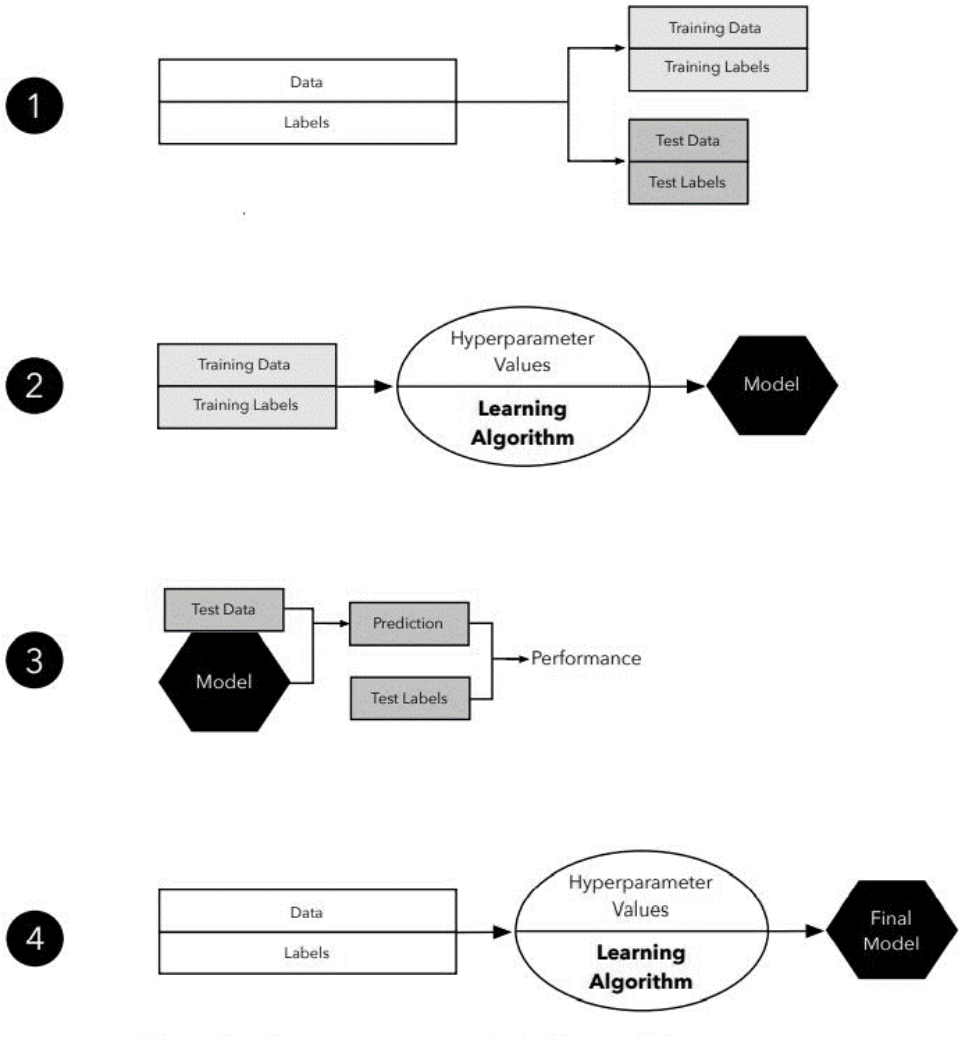
Sample: 3D visualization of random sport activity using coordinates in (x,y) axis. Left: z=altitude, right: z=speed

A-2.2C: SEQUENCE GENERATOR FUNCTION

```
import pandas as pd
import numpy as np

def get_sequences(data_path, entries, features):

    sequences = list()

    for entry in entries:
        # Read entry (csv file) to pandas dataframe for data processing
        df_data = pd.read_csv(data_path + entry, delimiter=';')

        # Set index to datatime object (one second intervals)
        df_data.index = pd.to_datetime(pd.Series(df_data.index))

        # Select features (given as function parameter)
        df_data = df_data[features].copy()

        # Replace string type of 'None' to np.nan
        df_data.replace('None', np.nan, inplace=True)

        # Transform values to numeric format
        df_data = df_data.apply(pd.to_numeric, errors='ignore')

        # Fills nan values linearly by context values use bidirectional method
        df_data.interpolate(method='linear',
                            limit_direction='both',
                            axis=0, inplace=True)
        # Removes all the rest of nan values where intepolate is not applicable
        df_data.fillna(0, inplace=True)

        # Take values and add them to the sequence list
        sequences.append(df_data.values.astype(float))
```

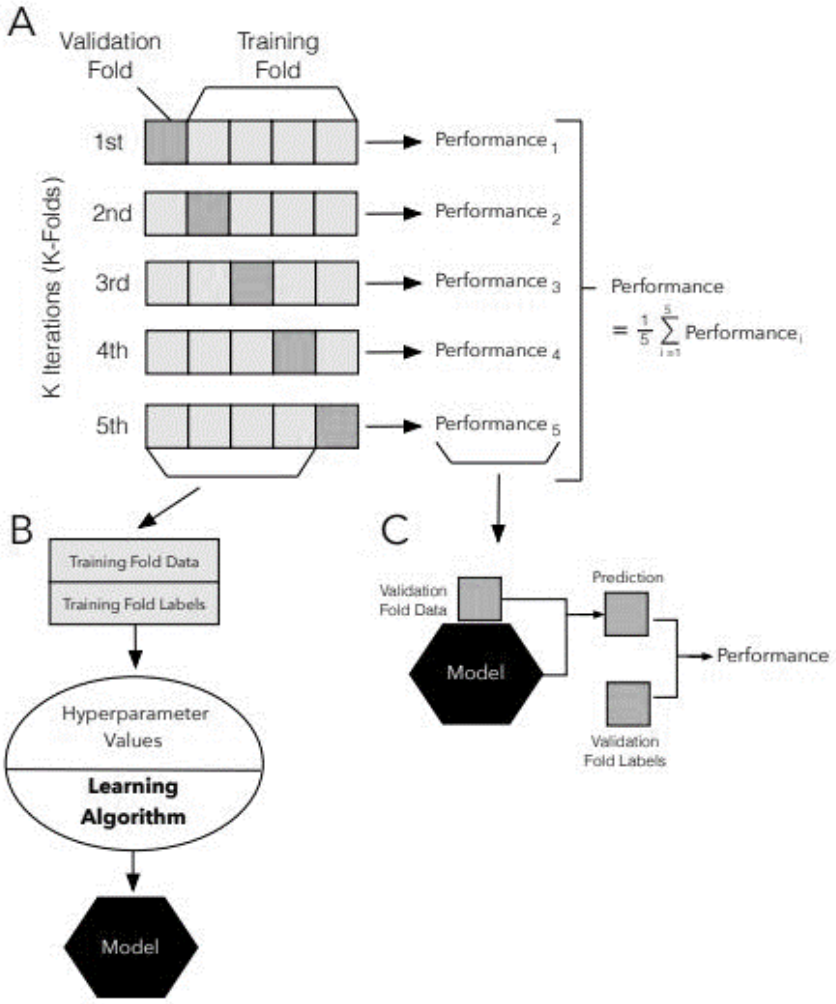
Code for generating sequences from the raw data.

A-3.1A: HOLDOUT VALIDATION METHOD



Holdout validation method (Raschka, 2018)

A-3.1B: K-FOLD CROSS-VALIDATION ALGORITHM



K-fold cross-validation algorithm (Raschka, 2018)

A-3.1C: OPTIMAL HYPERPARAMETERS FOR S-CML

Model	Hyperparameters	Value range or test values	Best hyper (std)	Best hyper (pca)
k-NN	n_neighbors	[1, 15]	3	3
	weights	{ 'uniform', 'distance' }	distance	distance
DT	criterion	{ 'gini', 'entropy' }	gini	gini
	max_depth	[3, 15]	7	7
	splitter	{ 'best', 'random' }	random	random
RF	n_estimators	{ 10, 20, 30 }	20	30
	max_features	{ 'auto', 'sqrt' }	auto	sqrt
	max_depth	{ 5, 10, 15, 20 }	15	15
	min_samples_split	{ 2, 5, 10, 20 }	5	5
NN	hidden_layer_sizes	{ (10, 30, 10), (20,), (50,) }	(50,)	(50,)
	activation	{ 'relu' }	relu	relu
	solver	{ 'sgd', 'adam' }	adam	adam
	alpha	{ 0.0001, 0.001, 0.01 }	0.01	0.0001
	learning_rate	{ 'constant', 'adaptive' }	constant	constant
NB	var_smoothing	logspace(0, -9, num=100)	0.00053367	0.043287613
LDA	shrinkage	{ 'auto', None }	None	None
	solver	{ 'lsqr', 'eigen' }	lsqr	lsqr
QDA	reg_param	{ 0.001, 0.01, 0.1, 1.0 }	0.01	0.1
	tol	{ 0.00001, 0.0001, 0.001, 0.01 }	1.00E-05	1.00E-05
	store_covariance	{ True, False }	TRUE	TRUE
SVM	C	{ 0.1, 1, 10, 100, 1000 }	1	1
	kernel	{ 'poly', 'rbf', 'sigmoid', 'linear' }	linear	linear
	gamma	{ 3, 2, 1, 0.1, 0.01 }	3	3
LR	C	logspace(-4, 4, num=50)	0.828642773	0.568986603
	penalty	{ 'l1', 'l2' }	l2	l2

Mode combinations of the best hyperparameters among 20 iterations by models. "Mode" means the most commonly appearing combination of parameters when conducting iterative hyperparameter optimization with different random state of the data division among train and test splits.

A-3.2A: TSC ALGORITHM PSEUDO CODES

Algorithm 1. buildTSF(A list of n cases length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of trees, k; the minimum interval length, p; the number of intervals per tree, r. (default $k \leftarrow 500$, $p \leftarrow 3$, and $r \leftarrow \sqrt{m}$)

Let $\mathbf{F} = (\mathbf{F_1} \dots \mathbf{F_k})$ be the trees in the forest
$i \leftarrow 1$
while $i < k$ **and** timeRemaining() **do**
  Let $\mathbf{S}$ be a list of n cases $(\mathbf{s_1} \dots \mathbf{s_n})$ with $3r$ attributes
  for $j \leftarrow 1$ to r **do**
    $b \leftarrow$ randBetween$(1, m - p)$
    $e \leftarrow$ randBetween$(b + p, m)$
    for $t \leftarrow 1$ to n **do**
      $s_{t,3(j-1)+1} \leftarrow$ mean$(\mathbf{x}_t, b, e)$
      $s_{t,3(j-1)+2} \leftarrow$ standardDeviation$(\mathbf{x_t}, \mathbf{b}, \mathbf{e})$
      $s_{t,3(j-1)+3} \leftarrow$ slope$(\mathbf{x}_t, b, e)$
  $\mathbf{F_i}$.buildTimeSeriesTree$(\mathbf{S}, y)$

Algorithm 1: Time Series Forest (TSF)

Algorithm 2. buildRISE(A list of n cases of length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of trees, k; the minimum interval length, p. (default $k \leftarrow 500$, $p \leftarrow$min$(16, m/2)$)

Let $\mathbf{F} \leftarrow < \mathbf{F_1} \dots \mathbf{F_k} >$ be the trees in the forest.
$i \leftarrow 1$
while $i < k$ **and** timeRemaining() **do**
  buildAdaptiveTimingModel()
  if $i = 1$ **then**
    $r \leftarrow m$
  else
    $max \leftarrow$ findMaxIntervalLength()
    $r \leftarrow$ findPowerOf2Interval(p, max)
  $b \leftarrow$ randBetween$(1, m - r)$
  $\mathbf{T}' \leftarrow$ removeAttributesOutsideOfRange$(\mathbf{T}, \mathbf{b}, \mathbf{r})$
  $\mathbf{S} \leftarrow$ getSpectralFeatures$(\mathbf{T}')$
  $\mathbf{F_i}$.buildRandomTreeClassifier$(\mathbf{S}, y)$
  updateAdaptiveModel(r)
  $i \leftarrow i + 1$

Algorithm 2: Random Interval Spectral Ensemble (RISE)

Algorithm 1. buildTSF(A list of n cases length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of trees, k; the minimum interval length, p; the number of intervals per tree, r. (default $k \leftarrow 500$, $p \leftarrow 3$, and $r \leftarrow \sqrt{m}$)

Let $\mathbf{F} = (\mathbf{F_1} \dots \mathbf{F_k})$ be the trees in the forest
$i \leftarrow 1$
while $i < k$ **and** timeRemaining() **do**
  Let $\mathbf{S}$ be a list of n cases $(\mathbf{s_1} \dots \mathbf{s_n})$ with $3r$ attributes
  for $j \leftarrow 1$ to r **do**
    $b \leftarrow$ randBetween$(1, m - p)$
    $e \leftarrow$ randBetween$(b + p, m)$
    for $t \leftarrow 1$ to n **do**
      $s_{t,3(j-1)+1} \leftarrow$ mean$(\mathbf{x}_t, b, e)$
      $s_{t,3(j-1)+2} \leftarrow$ standardDeviation$(\mathbf{x_t}, \mathbf{b}, \mathbf{e})$
      $s_{t,3(j-1)+3} \leftarrow$ slope$(\mathbf{x}_t, b, e)$
  $\mathbf{F_i}$.buildTimeSeriesTree$(\mathbf{S}, y)$

Algorithm 3: Shapelet Transform Classifier (STC)

Algorithm 4. HIVE-COTE classify(A test case $\mathbf{x}$)

Return: prediction for case $\mathbf{x}$

Parameters: A set of classifiers $< M_1, \dots, M_k >$, an exponent α, a set of weights w_i, and the number of classes c

$\{\hat{p}_1, \dots, \hat{p}_c\} = \{0, \dots, 0\}$
for $i \leftarrow 1$ to k **do**
  for $j \leftarrow 1$ to c **do**
    $\hat{q_j} \leftarrow \hat{p}((y = j|M_i, \mathbf{x})$
    $\hat{p}_j \leftarrow \hat{p}_j + w_i^{\alpha} \cdot \hat{q}_j$
return $argmax_{j=1\dots c}\hat{p}_j$

Algorithm 4: HIVE-COTEv1.0

Algorithm 5. WEASEL+MUSE(A list of n cases of length m with dimension d, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the word length l, the alphabet size α, the maximal window length w_{max}, mean normalisation parameter p, equi-depth or equi-frequency binning b

Let $\mathbf{H}$ be a collection of n histograms $\mathbf{h}$
Let $\mathbf{B}$ be a matrix of l by α breakpoints found using b
$\mathbf{X}' \leftarrow$ addDerivativesAsDimensions($\mathbf{X}$)
for $i \leftarrow 1$ to n **do**
  for $k \leftarrow 1$ to $2d$ **do**
    for $w \leftarrow 2$ to w_{max} **do**
      for $j \leftarrow 1$ to $m - w + 1$ **do**
        $\mathbf{o} \leftarrow x'_{i,j,k} \ldots x'_{i,j+w-1,k}$
        $\mathbf{q} \leftarrow$ DFT(o, w, p) { $\mathbf{q}$ *is a vector of the complex DFT coefficients*}
        $\mathbf{r} \leftarrow$ SFAlookup($\mathbf{q}$, $\mathbf{B}$)
        $pos \leftarrow$index($\mathbf{k}$, $\mathbf{w}$, $\mathbf{r}$)
        $h_{i,pos} \leftarrow h_{i,pos} + 1$
$h \leftarrow \chi^2(h, y)$ { *feature selection using the chi-squared test* }
fitLogistic(h, y)

Algorithm 5: WEASEL-MUSE (MUSE)

A-3.2B: PARAMETER SETUP FOR TSC MODELS

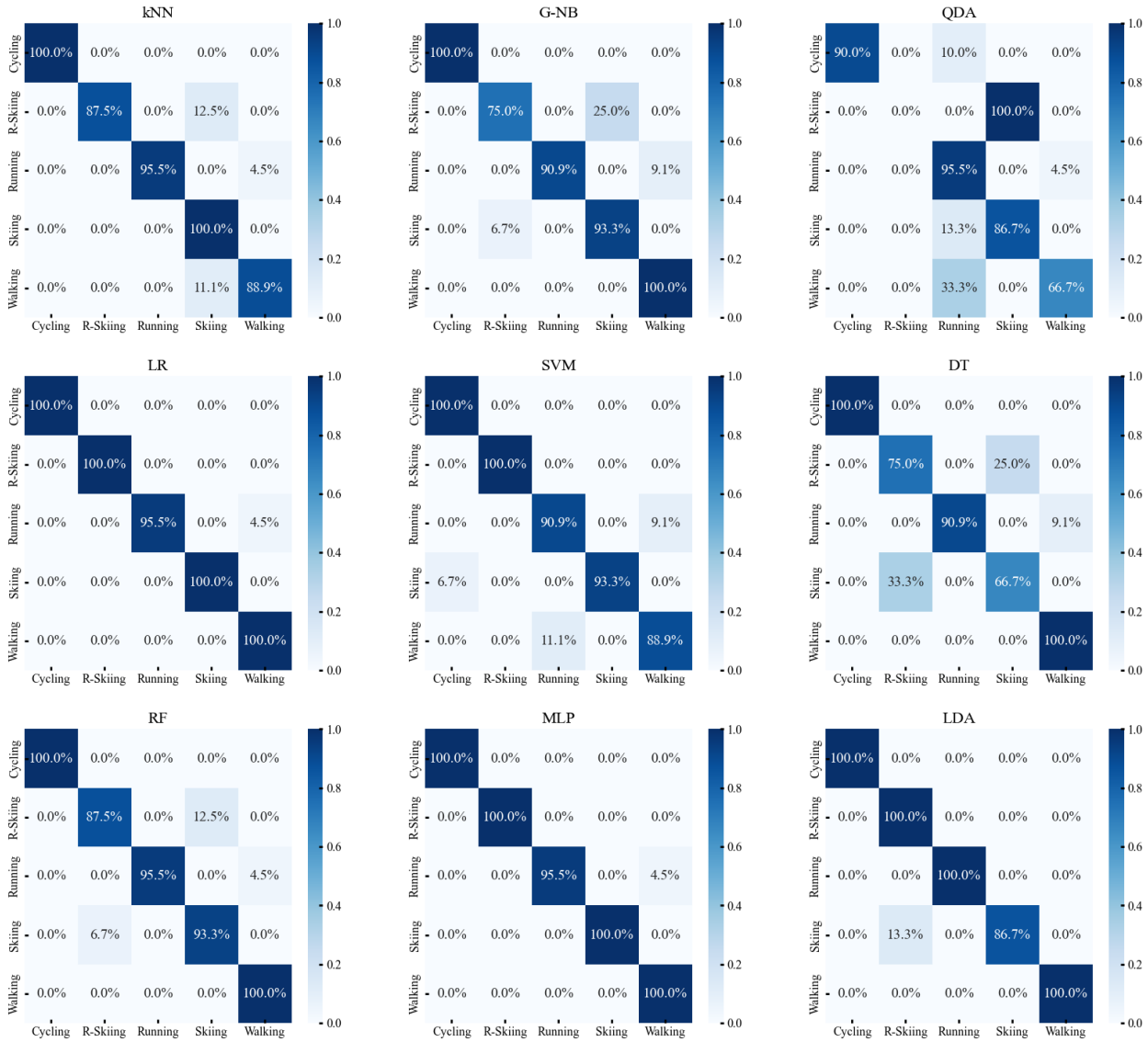
Parameter name	TSF	STSF	RISE	RIC	STC	kNN-TS	CTSF	WEASEL	MUSE
min_interval	3		16						
n_estimators	200	200	500				100		
n_jobs	1	1	1	1	1			1	1
acf_lag			100						
acf_min_values			4						
max_interval			0						
n_intervals				100					
batch_size					100				
contract_max_n_shapelet…					inf				
n_shapelet_samples					10000				
save_transformed_data					FALSE				
time_limit_in_minutes					0				
transform_limit_in_minutes					0				
algorithm						brute			
distance						dtw			
leaf_size						30			
n_neighbors						1			
pass_train_distances						FALSE			
weights						uniform			
bootstrap							FALSE		
min_impurity_decrease							0		
min_samples_leaf							1		
min_samples_split							2		
min_weight_fraction_leaf							0		
oob_score							FALSE		
verbose							0		
warm_start							FALSE		
alphabet_size								2	4
anova								TRUE	TRUE
bigrams								TRUE	TRUE
binning_strategy								information-gain	
feature_selection								chi2	chi2
p_threshold								0.05	0.05
support_probabilities								TRUE	TRUE
window_inc								2	4
use_first_order_differences									TRUE
variance									FALSE

Default model hyperparameter setup for U-TSC and M-TSC

A-3.4A: CLASSIFICATION FUNCTION

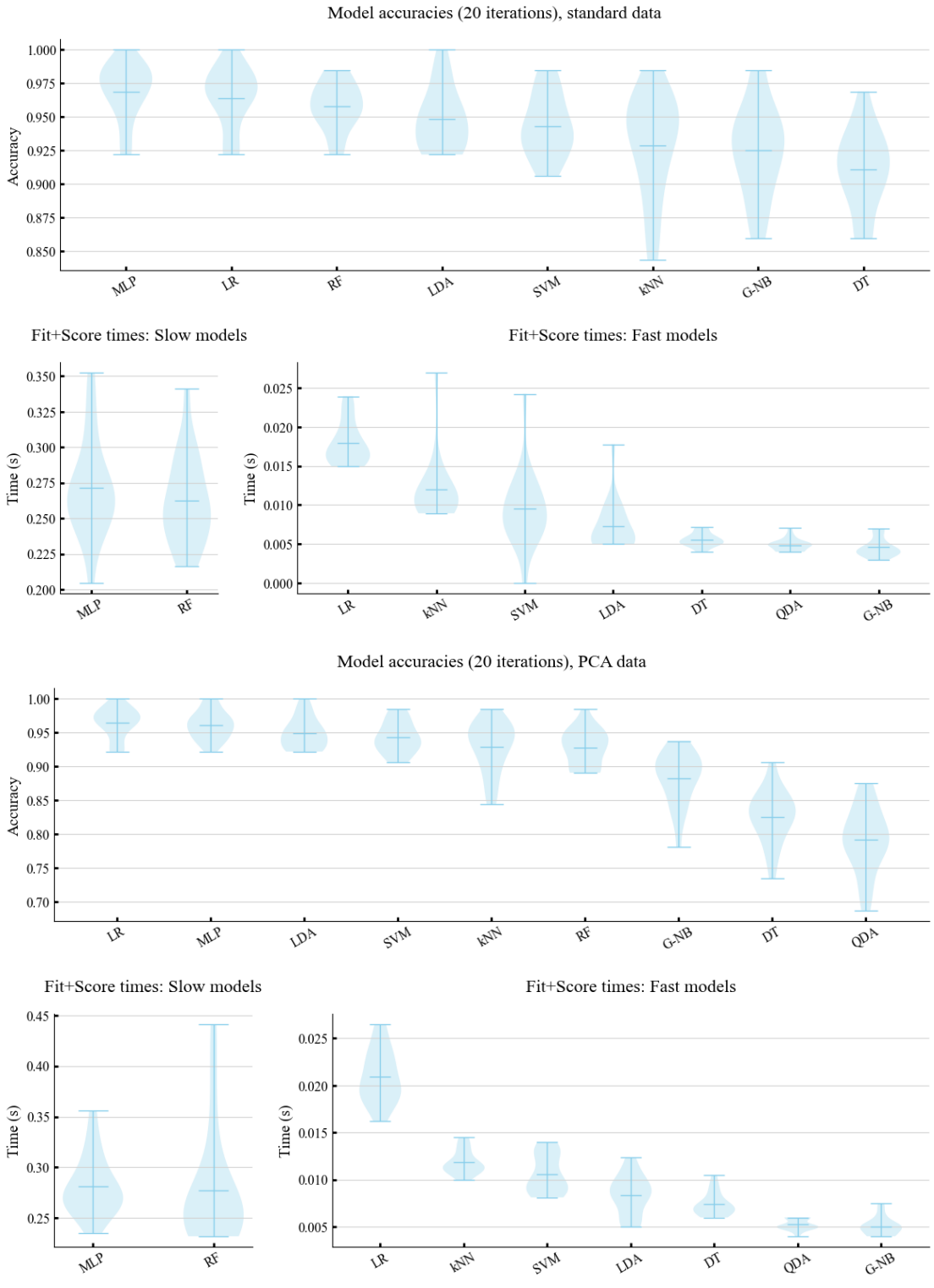
```
def classify(classifiers, clf_type, X_train, X_test, y_train, y_test, results, preds, iters):
    # Pandas dataframe for results
    if results is None:
        results = pd.DataFrame(columns=['Classifier','Type','Train(t)','Test(t)'])
        predictions = pd.DataFrame()
    # Create column names for scores according to the number of iterations.
    score_col_names = create_col_names('Score_', iters)
    # Create progress bar with non-default styles
    progress_bar = tqdm(classifiers.items(), ncols=100, colour='#87ceeb', file=sys.stdout)
    # Iterate for each TSC models.
    for name, clf in progress_bar:
        # Includes bold text printing.
        progress_bar.set_description('Processing \033[1m ➥%s \033[0m' % str(clf))
        score_row = pd.DataFrame(data={'Classifier':name,'Type':clf_type,}, index=[0])
        # Insert score colums for each iteration.
        for i in range(1,iters+1):
            score_row[score_col_names] = 0
        # Initialize best score variable
        best_score = 0
        # Iterate classification block 'iters' time
        for iter in range(1,iters+1):
            start = time()                          # Start timing the model
            clf.fit(X_train, y_train)               # Fit the data
            train_time = time() - start             # Stop train timer, save value
            start = time()                          # Start test timer
            y_pred = clf.predict(X_test)            # Predictions
            score = accuracy_score(y_test, y_pred) # Model scores
            score_time = time()-start               # Stop test timer, save value
            # Set values (note: time data will be overwritten in each iteration)
            score_row['Train(t)'] = train_time
            score_row['Test(t)'] = score_time
            score_row[score_col_names[iter-1]] = score
            # Among iterations, we could take only one f1 and roc.
            # We store them for the best accuracy execution on the following code block:
            if score > best_score:
                # Store predictions for other than multivariate models
                predictions[name] = y_pred
                # F1 and ROC-AUC scores
                score_row['f1'] = f1_score(y_test, y_pred, average='micro')
                # Propabilities are not available for sklearn models except tree models
                # Therefore, ROC-AUC is not calculated for those
                if clf_type not in ['sklearn']:
                    y_pred_proba = clf.predict_proba(X_test)
                    score_row['roc-auc'] = roc_auc_score(y_test, y_pred_proba,
                                                         average='weighted',
                                                         multi_class='ovr')
                else:
                    score_row['roc-auc'] = np.nan
                # Update best score variable
                best_score = score
        # Save model scores to the result table
        results = pd.concat((results, score_row), axis=0, ignore_index=True)

    print('Classification done for ' + clf_type + '\n')

    return (results, predictions)
```

A-4.1A: CONFUSION MATRICES (DEFAULT)

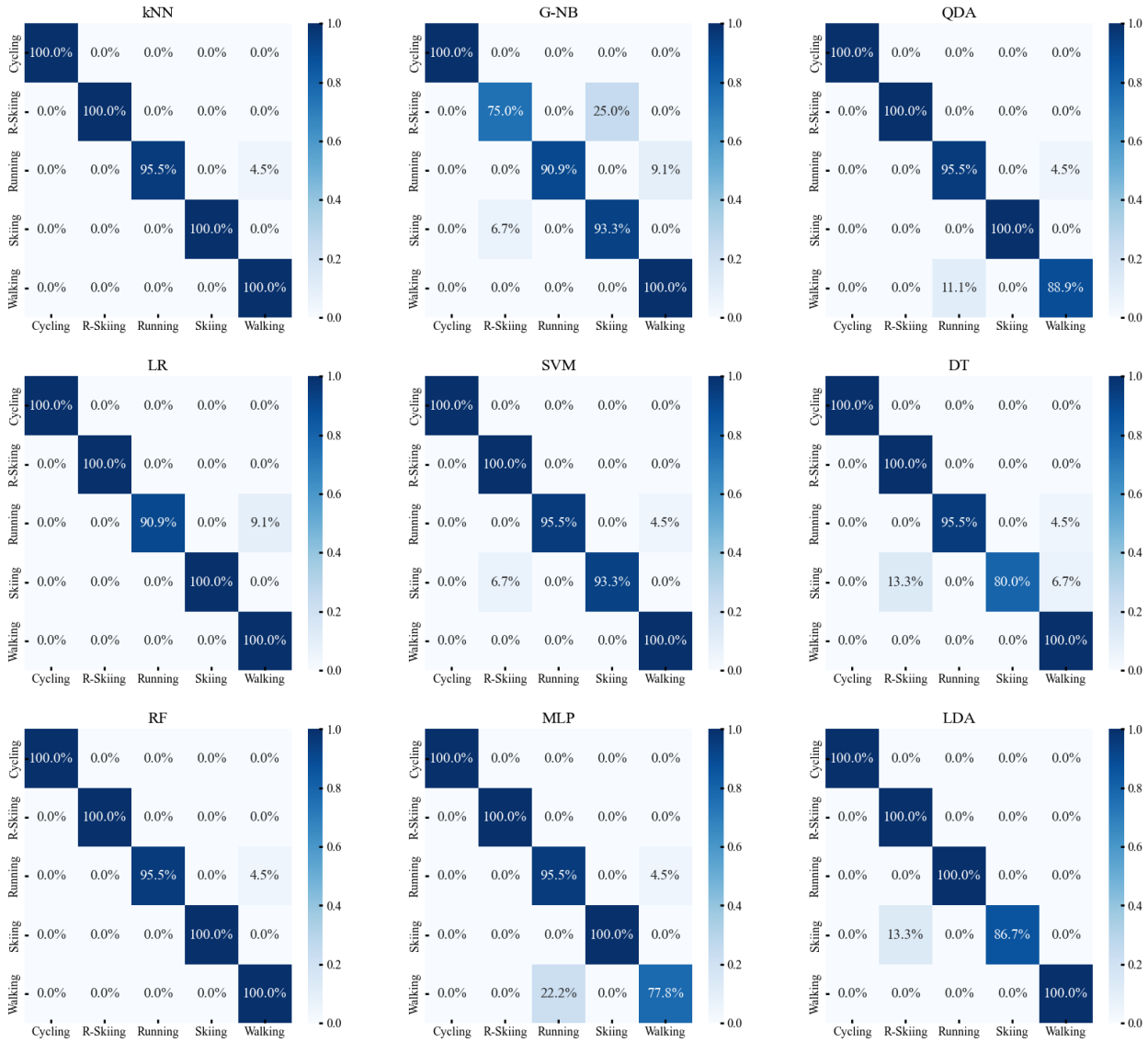


Appendix 4.1A: Confusion matrixes of classification with default hyperparameters in standard data.

A-4.1B: VIOLIN PLOTS OF THE CLASSIFICATION RESULTS (OPTIMAL)



A-4.1C: CONFUSION MATRICES (OPTIMAL)



Appendix 4.1C: Confusion matrixes of classification with optimal hyperparameters in standard data.

A-PC: PROJECT CODES AND DATASETS

Github

SAC project repository: https://github.com/JABE22/MasterProject

TCX-CSV file converter: https://github.com/JABE22/TCX-CSV_File_converter.git

Kaggle

Jupyter files with preview:

Part (1/3): https://www.kaggle.com/code/jarnomatarmaa/masterprojectcode-cml

Part (2/3): https://www.kaggle.com/code/jarnomatarmaa/masterprojectcode-tsc

Part (3/3): https://www.kaggle.com/code/jarnomatarmaa/masterprojectcode-sac-analysis

Dataset

Original data for SCM: https://data.world/jarnoma/activities

Original data for TSC: https://www.kaggle.com/datasets/jarnomatarmaa/sportdata

Interlaced TSC data: https://data.world/jamasoftwares/outdoor-sport-activities

Related dataset

Important note: this is not the dataset used in this project, but as a reference material here.

UCI ML Repository: https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities

Project for dataset: https://github.com/JABE22/sports_activities_dataset

A-TE: TEST ENVIRONMENT AND DEVICE

Device	**Lenovo Yoga 920-13IKB**
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (1.80 GHz)
Installed RAM	8,00 GB
System type	64-bit operating system, x64-based processor
Development environment	**Visual Studio Code, Jupyter extension** Version: 1.74.3 (user setup) Commit: 97dec172d3256f8ca4bfb2143f3f76b503ca0534 Electron: 19.1.8 Chromium: 102.0.5005.167 Node.js: 16.14.2 V8: 10.2.154.15-electron.0 OS: Windows_NT x64 10.0.22621 Sandboxed: No

A-DEVS: DATA RECORDING DEVICES

Devices used in the recording of sport activities. All the data has been collected by these two devices.

***Garmin Forerunner 920XT and HRM-Run sensor**. A chest strap features a module and contact patches that can detect and measure your pulse via electric signals emitted by your heart. This information is then transmitted to a connected device that records the data during an activity.*

***Garmin vivosport.** Optical sensor measures heart rate from the wrist. Optical heart rate sensors work by using lights that can measure small changes in blood volume in the capillary layer just below the epidermis (outermost of the three layers that make up the skin) as blood flows through the wrist area. This information is stored for either all-day heart rate analysis or recorded activity analysis on the watch.*

Garmin, available at *https://support.garmin.com/en-US/?faq=2yWINctFF76LmTZ3aw2Wt9*