Vrije Universiteit Amsterdam

X 400459

Master Project Business Analytics

Activity Context Detection during Smartphone Keyboard Interactions: A Machine Learning Approach

Author:
Kostantinos Sakellariou

Type:
Internship Report

External Supervisor:
Dr. Giovanni Licitra

Internal Supervisors:
prof. Dr. Eliseo Ferrante
prof. Dr. Jacub Tomczak

Vrije Universiteit Amsterdam

X 400459

Master Project Business Analytics

Activity Context Detection during Smartphone Keyboard Interactions: A Machine Learning Approach

Author:
Kostantinos Sakellariou

Host Organization:
Neurocast B.V
Amsterdam office
De Entree 234, 1011 EE
Amsterdam, The
Netherlands

University:
Vrije Universiteit
Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

Date:
November 2021

Abstract

There is a continuous demand of the modern lifestyle, the use of smartphones or smartwatches or in general the use of any piece of technology, in order people to monitor and potentially improve their health condition. Self-tracking from devices can generate valuable data, but the main issue is that the data are collected in a real-world environment; hence they inherently carry noise due to the underlying activity done while interacting with the smartphone. For example, keystroke dynamics can be utilised as a digital biomarker in relation to clinical outcomes used by clinicians to assess, e.g. fine motor skills. However, the typing behaviour is likely affected by what the person is doing while typing. This research project investigates the boundary conditions within a given typing session, and this happens in three different levels: investigate if the person who is typing is walking or performing any other non-motion activity, investigate if the person is typing with one or two hands and finally if the person is typing with the thumb or the index finger. Collecting the data and following a machine learning workflow (exploratory data analysis, feature engineering and selection, data preparation, machine learning models) yielded the following results per target: for target activity (distinguish between walking or non-motion activity), the best performing model achieved an F1 score=0.955 in cross-validation and on the hold-out set an F1 score=0.927. For the handedness target, the cross-validation F1 score on the training set was 0.835, and on the hold-out set 0.818, and for the finger dominance target, the best model achieved an F1 score=0.92 on the training set with cross-validation and an F1 score=0.909 on the Hold-out set. For the models to predict based on the typing sessions, they leveraged a combination of features coming both from the keystroke and the sensor dataset. These findings suggest that the combination of keystroke and sensor features have sufficient predictive power in order to distinguish between the aforementioned boundary conditions, which is beneficial if one wishes to explain the variability coming from the keystrokes; however, further research is still to be conducted in the direction of using these results in the healthcare context.

Acknowledgements

After six constructive months working on the master project with the title "Activity Context Detection during Smartphone Keyboard Interactions: A Machine Learning Approach", finally it is the moment to express my unlimited gratitude to those who supported and contributed to this research project.

Foremost, I would like to express my special thanks to my supervisor from the host organization, Giovanni Licitra, who believed in me and gave me the chance to excel and improve myself. Also, I would like to thank him for his detailed feedback, advice and support throughout the six-month internship period.

Furthermore, I would like to thank the Vrije University supervisors, my first supervisor Eliseo Ferrante and my second reader Jacub Tomczak for their help and contribution whenever it was needed. Also, I would like to express my gratitude to Neurocast B.V, the host company, where I was able to work at their offices, meet great people, and use their resources and technology, which were key in order to successfully complete this research project.

Last but not least, I would like to express my heartful gratitude to Georgia for being by my side and supporting me throughout this internship and for her contribution to the data collection. Alongside, I would like to thank my parents for being constantly supportive and helpful.

Special thanks to all those people who participated and spent their valuable time and effort to help me collect the data that were needed in order to complete this research-internship project.

Contents

1 Introduction

1.1 Research Context

According to [24], biometric data are physical or behavioral features that are unique to each person and can be used as a means of individual validation for security purposes. Physical biometric features can include, for example, finger prints, iris scans, and face scans, which are unlikely to change except through physical damage. Behavioral features are able to change based on the variable state of a person and can include keystroke dynamics, gait, voice, and other personal identifiers from which an overall personal profile can be developed. The biometrics of keystroke dynamics are based on the assumption that different people have different typing mannerisms and that these neuro-physiological factors are reflected in the data of that individual, leading to a "typing signature" of a person at any given time. Neurocast B.V. [16] is a company that operates in this context.

Neurocast is a med-tech company, which specializes in passively monitoring patients. The company aims to leverage the interaction with smartphone technology, such as keystroke dynamics and sensor data, in order to produce valuable insights regarding a person's mental, physical and emotional status. Because people with chronic illnesses do not want to be confronted with their illness and actively monitored, doctors and researchers cannot come up with valuable results; hence Neurocast introduced its technology to collect data in a entirely unobtrusive manner, with privacy embedded by design and available for everyone. Through their platform and technology Neurocast obtains a complete picture of a patient's clinical data by collecting a consistent flow of real-world data. Neurocast's mission is to turn everyday digital interactions into clinically approved outcomes, enabling doctors and researchers to measure individual patients' performance in daily life passively.

1.2 Research Objective

In this era, the modern lifestyle demands the use of technology as part of our life. People who carry a smartphone and wear a smartwatch can potentially gather a significant amount of data, that can be used, e.g., to monitor and improve the person's health condition. Self-tracking from devices can generate useful data, but the main issue is that the data are collected in a real-world environment, hence they inherently carry noise due to the underlying activity done while interacting with the smartphone. This is where machine learning can help us understand better the data. Using machine learning techniques, one can capture meaningful insights from the data and turn them into useful suggestions or predictions. In this work, the focus will be on data collected passively during a typing session via an application developed by Neurocast B.V., namely, Neurokeys [17].

According to [12], the analysis of press and release keyboard interactions - keystroke dynamics - allows the identification of typing behaviour (e.g. amount,

speed and error rate of typing) in a real-world setting which can reflect motor and non-motor functioning. It is hypothesised that keystroke dynamics can be utilised as a digital biomarker in relation to clinical outcome used by clinicians to assess e.g. fine motor skills and fatigue. However, since the typing behaviour is likely affected by what the person is doing while typing, the goal is to infer the boundary conditions within a given typing session. By boundary condition it is meant e.g.:

- infer if the person is typing with one or two hands;
- infer if the person is walking or sitting;
- infer if the person is typing with the thumb or the index finger.

The knowledge of such boundary conditions can be used to explain part of the variance coming from the Keystroke Dynamics, while the remaining information can be linked to a specific clinical outcome. Furthermore, it will be investigated if it is possible to flag the typing session as an anomaly whenever someone else is typing in the smartphone. In this way, the anomalous typing session can be discarded prior to any clinical assessment using the Neurocast technology.

1.3 Literature Review

This subsection provides the primary research in the broader research context of this project, as this specific research has not been developed yet by any other researcher or research team.

Lam [12] in his study takes advantage of the Neurokeys App [17] in order to collect data and investigate the keystroke dynamics in the field of Multiple Sclerosis. The aim of the study was to determine the reliability and validity of keystroke dynamics to assess clinical aspects of Multiple Sclerosis. Keystroke dynamics were reliable, distinguished patients and controls, and were associated with clinical disability measures.

Twose et al. [24] in his study aim to test the feasibility of using Non-Linear Time Series Analysis within Keystroke Dynamics. The results showed a daily change in Keystroke Dynamics for all users but only clinically relevant changes in the population with Multiple Sclerosis.

The work by Voicu et al. [25] takes advantage of a custom smartphone app available for Android devices, and the smartphone sensors (accelerometer, gyroscope and gravity sensors) in order to perform activity recognition using a deep learning technique to classify the raw data in six different activities, namely walking, sitting, standing, running, walking upstairs and downstairs. It is shown how human activity recognition can be achieved by using sensors available on a smartphone.

He et al. [26] utilize the Built-In Kinematic Sensors of a Smartphone in order to perform physical activity recognition. The sensors included in their work were the triaccelerometer, the gyroscope and the magnetic sensor. The results

indicated that the accelerometer was significant to physical activity recognition, while gyroscope was effective to recognize the change of body posture. In this work, the accelerometer and the gyroscope sensors are utilized in order to make accurate predictions about the activity of the smartphone users.

Dagum [3] in order to identify digital biomarkers associated with cognitive function, analyzed human–computer interaction of smartphone use in 27 subjects. Utilizing an app that worked on the background and captured tactile user activity that included swipes, taps, and keystroke events, collectively termed human–computer interactions (HCI). It is shown that one using passively acquired data during daily use of a smartphone, can generate digital biomarkers correlated with gold-standard neurocognitive tests.

1.4 Summary Results

The outcome of this research project is that using machine learning techniques, one can predict the boundary conditions using the suggested features. The results of the research were promising enough to conclude about the practical aspect of the models. The key performance metric that is used to measure the performance of each model is F1 score, which is explained in Section 3.6. Briefly, for every target the results are summarized as followed:

- **Activity:** The results that are achieved for target activity are: F1 score 0.927 and AUC of 0.92. The models are certain enough in predicting if the person is performing a motion or non-motion activity.
- **Handedness:** The results that are achieved for target handedness are: F1 score 0.818 and AUC of 0.80. The models are certain enough in predicting if the person is typing with two hands.
- **Finger Dominance:** The results that are achieved for target finger dominance are: F1 score 0.909 and AUC of 0.84. The models are certain enough in predicting if the person is typing with the thumb.

1.5 Thesis Structure

The thesis is organized in the following three main sections:

- **Section 2** refers to the study overview and explains in detail the set-up of the project, the study design, the data collection, the three different datasets that are introduced, the creation of the Hold-out set and assumptions and limitations of the project.
- **Section 3** addresses the methodology of the research project and includes feature engineering, feature selection, exploratory data analysis of the datasets, the models used to make the predictions, the machine learning approaches, the required data preparation and the evaluation metrics.

- **Section 4** of the thesis is the results Section, where the results are explained in detail for every target separately, and error analysis is conducted, followed by feature importance.
- **Section 5 and 6** discuss the limitations, assumptions and recommendations of this research project, and the final section is the conclusion, where the research question is answered.

2 Study Overview

The current section will address the study design: a description of the populations and how and when the data were collected; keystroke data acquisition: an introduction to the Neurokeys technology; and the ethical approval of the study.

2.1 Study Design

The data have been collected through the Neurokeys app [17], which is developed to measure health status through regular typing on the smartphone. All keyboard interaction data were remotely collected throughout the internship. The data from the participants were collected from June 2021 to September 2021, and all the participants were considered healthy subjects. The age of the participants is concentrated between 26 to 34, except for one user who is at the age of 55, and the data was collected from iOS devices only. Table 1 summarizes the information of the research internship.

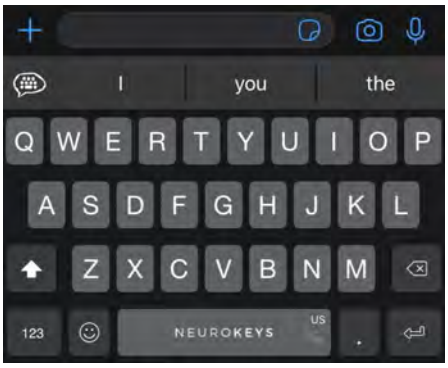
Table 1: Summary of the study design

App	OS	Participants	Age	Condition	Duration
Neurokeys	iOS	9	26-55	Healthy	4 months

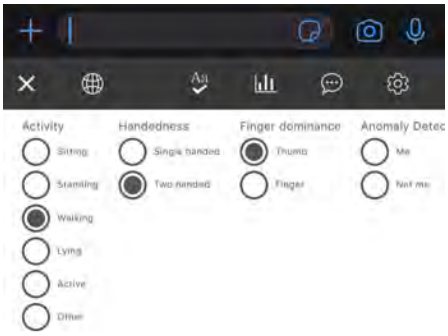
2.2 Data Collection

A smartphone app (Neurokeys, Neurocast B.V., Amsterdam) was developed for Android and iOS to measure health status through regular typing and via sensors, GPS, and questionnaires. The Neurokeys keyboard was installed on the participants' smartphones and replaced the default. At the beginning of typing session, the participant labels the typing session. Chooses the activity category between lying, sitting, standing, walking or other, the finger dominance category between thumb and finger, the handedness category between single-handed or two-handed and the anomaly detection category between "me" and "not me". The labelling of the typing session can also be chosen after the participant is finished typing. During regular typing, keyboard interactions of interest were logged and timestamped in the background: alphanumeric keys, backspaces, space bars and punctuation keys. Based on these timestamped key types, the manner and rhythm of typing can be discerned by analysing keystroke features. Keyboard interactions and activity tracking data were continuously collected and stored per typing session, defined as one successive period of activation followed by inactivation of the keyboard. When a typing session starts, keystroke data, labels and sensor data from the previous session are sent and removed from the smartphone. The final dataset is constructed by merging the three different types of datasets. The collection of data through Neurokeys did not require

additional action from the participants. Finally, note that if the labels are not provided for a given typing session, underlying data are discarded a priory. Figure 1a and Figure 1b show a screenshot of the Neurokeys keyboard and the smart panel questionnaire.



(a) The Neurokeys Keyboard



(b) The labeling options in the Neurokeys app

2.2.1 Target Dataset

Throughout this work, supervised machine learning techniques are leveraged. In short, such an approach is defined by its use of labelled datasets to train algorithms to classify data and/or predict outcomes. In our case, the labels coming from the smart panel questionnaire are used as a target to train mathematical models. Once the data are collected, one can request them from the cloud server using customized API, and they are returned as pandas data frame. The labels are returned sorted by timestamp; hence the labels of a typing session are located in different rows of the data frame. Furthermore, `user_ID` and `session_ID` are provided in order to discern data among users and instances, respectively. An example of the target dataset is shown below in Table 2. Note that the data are utterly artificial due to privacy reasons; however, they are representative of the original.

Table 2: The labels dataset

session_ID	timestamp	user_ID	activity	handedness	finger dominance	anomaly detection
MFJNVDI9469	2021-06-02 07:57:23	2300	walking	single_handed	thumb	me
PFVMF245607	2021-06-28 10:24:58	280	standing	two_handed	thumb	me
NVUREJ23498	2021-08-10 16:45:20	1458	lying	single_handed	finger	not_me
JNCFVJFU245	2021-08-18 22:50:03	575	sitting	two_handed	finger	me
BUITN694786	2021-09-24 04:12:48	834	other	single_handed	thumb	not_me

Before preprocessing, the labels dataset was of shape (9403, 19), where 9403 are the rows, and 19 are the data frame columns. After a preprocessing procedure, the total amount of labels is 2224, which can also be seen as the dataset's upper bound value valid for model training and validation.

2.2.2 Feature Dataset: Keystroke Data

The keystroke dataset is the dataset that contains all the necessary information about the typing behaviour of the participant in a given typing session. When such a data source is requested from the cloud server, a pandas data frame that contains all the keystroke features aggregated via different summary statistics (e.g. mean and standard deviation) per typing session and per user is provided. The keystroke dataset is returned sorted by timestamp, and the features in this dataset are viewed as aggregations of the current typing session. The essential features of the keystroke dataset are described in [12] and the short definition is below.

- **Hold Time:** The time a key is pressed;
- **Flight Time:** The time between a key being released and the next key press;
- **Press Press Latency:** The time between successive key presses;
- **Release Release Latency:** The time between successive key releases;
- **Correction Duration:** The amount of time during which a user is correcting a mistake.

An instance of the keystroke dataset is shown below in Table 3. Note that the data are entirely artificial due to privacy reasons; however, they are representative of the original.

Table 3: The keystroke dataset. All features shown in this table are aggregations of the average time (in milliseconds) for a specific typing session

timestamp	**user_ID**	**session duration**	**correction duration**	**release release latency**	**press press latency**
2021-06-02 07:57:23	2300	6429.70	633.66	349.68	632.40
2021-06-28 10:24:58	280	2456.90	450.89	325.81	546.76
2021-08-10 16:45:20	1458	23048.19	232.67	678.12	347.34

Note that all the features mentioned above are expressed milliseconds. In addition, for each feature are calculated five summary statistics (i.e. vectors): mean and median (indicators of central tendency), standard deviation (SD; an indicator of dispersion) and minimum and maximum (indicators of range). Other statistics are also calculated and added as vectors of the feature as skew and kurtosis. Furthermore, features that count all the events in the typing session, the word length and the emojis are also included in the dataset. Figure 2 shows and explains schematically the keystroke features that were derived from timestamped keyboard interactions.

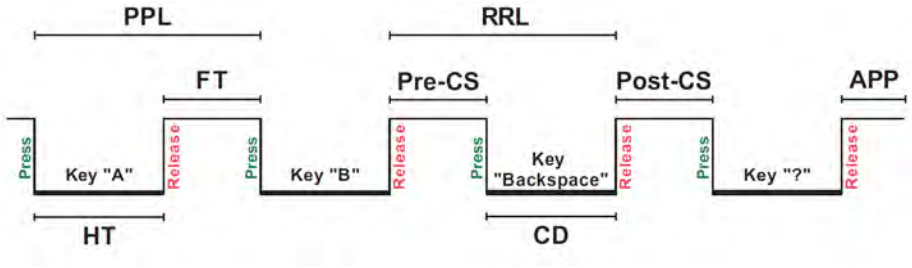


Figure 2: Schematic representation of the definition and aggregation of the timing-related keystroke features. Image courtesy of Lam et al. [12]

In the end, the original keystroke dataset prior to merging with the target dataset contains 17543 samples and 388 features. These features are subsequently combined in composites score where each cluster is created based on similarities observed from both a conceptual and a statistical point of view.

2.2.3 Feature Dataset: Sensor Data

The sensor data are collected within the typing session and can provide a significant insight regarding the user's motion while typing. The sensor dataset is divided into two subsets: the gyroscope and the accelerometer. The sensor dataset consists of (x, y, z) directions of accelerometer and gyroscope signals. At every typing session, signals are picked up at a frequency of 30Hz. However, the data is re-sampled at 15Hz to avoid mismatches between accelerometer and gyroscope samples. Therefore, sensors are not guaranteed to pick up signals flawlessly at 30Hz. Moreover, a Butterworth filter is applied to smooth the signal and increase the signal-to-noise ratio. Each of these datasets contains information about the device's movement when the participant is typing. A brief explanation can be found below.

- **Gyroscope:** A gyroscope allows us to keep track of the phone orientation. The ability of the gyroscope to accurately measure the orientation of an object is used in this work to capture the smartphone's orientation.
- **Accelerometer:** An accelerometer allows us to keep track of the phone's acceleration. From a standstill position changing to any velocity, the accelerometer can measure the object's acceleration; in this work, the smartphone acceleration.

When this dataset is requested from the cloud server, it is returned as pandas data frame containing all the built-in Neurocast's data pipeline features. The requested data are returned as aggregation per typing session, meaning that a data sample is composed of summary statistics relative to a specific user's motion on a given typing session. The sensor dataset is returned sorted by timestamp, and the features in this dataset are viewed as aggregations of the

current typing session. The essential features of the sensor dataset are described below. A feature instance of the gyroscope dataset is shown below in Table 4 and an instance of the features of the accelerometer dataset is shown in 5. Note that the data are entirely artificial due to privacy reasons; however, they are representative of the original.

Table 4: The gyroscope dataset

timestamp	user_ID	pitch	yaw	roll	omega
2021-07-07 09:52:34	2300	-0.599	0.264	0.245	3.234
2021-08-19 16:54:58	280	-1.109	0.361	-0.970	6.235
2021-08-14 21:36:09	1458	-0.183	0.086	-0.149	3.914

Table 5: The accelerometer dataset

timestamp	user_ID	x_filtered	y_filtered	z_filtered
2021-07-07 09:52:34	2300	-0.464	-0.831	0.359
2021-08-19 16:54:58	280	-0.356	-0.873	0.389
2021-08-14 21:36:09	1458	-0.574	-0.776	0.280

The features obtained from the movement of the participant and the device during the typing session are explained and defined as follows:

- **Pitch, Yaw, Roll:** Specify the rotation degrees around 3 axis. Unit ($rads$)
- **Angular velocity :** Specify the rate of change of the angular velocity of an object along 3 axis of its own local coordinate system. Unit rad/s
- **Angular acceleration:** Specify the rate of change of the angular acceleration of an object along 3 axis of its own local coordinate system. Unit rad/s^2
- **Accelerometer Filtered (x, y, z):** Specify the acceleration (rate of change of the velocity) of an object along 3 axis of its own local coordinate system. The unit is m/s^2

2.3 Creating a Hold-out Set

Creating a hold-out set is a prevalent practice in machine learning projects. This hold-out set, also named the test set, will be used only to evaluate the final models' performance because this is a way to avoid any bias coming from the data that the models have already seen and trained. Furthermore, two main reasons for creating a hold-out set are to prevent overfitting the training set and data leakage, i.e:

- **Overfitting** refers to a model that fits the training set in extreme level. That occurs when the model learns all the details of the training set and is not able to generalize this performance into unseen data [11].
- **Data leakage** refers to information used outside the training set and is used to construct the models. Data leakage can harm the predictive power of the models and yield to untrustworthy models predicting invalid estimations on the test set [9].

Due to the high imbalance of the sizes between the data collected from the participants and the reasons above, a hold-out set containing data from 7 users provides only 18% of the final dataset and the training set containing the rest 82% of the data was created. Thus, given the data of two participants, the approach is to predict the boundary conditions of the rest without any previous knowledge of those data points. The split between the dataset is summarized in Table 6

Table 6: The splitting of the dataset

	Training Set	**Hold-out Set**
Samples	1841	383
# of Users	2	7

2.4 Assumptions & Limitations

The most critical assumption is that each participant has correctly performed the labelling of the typing sessions; in other words, the participants have labelled all the typing sessions honestly and did not conduct any mislabeling by purpose (or unintentionally), which could lead to any mislabeling to a later stage in false evaluation.

Another assumption comes from the participants' demographics, where it is assumed that all participants are healthy and their typing behaviour did not change throughout the entire experiment under any circumstances. On top of that, the range of age of the participants was relatively small, leading to potential data drifting when predictions are made with people outside this range.

Finally, the study has been carried out purely on a healthy population; consequently, the machine learning models derived with such dataset might not be generalizable for any other type of population, e.g. a population affected by a specific chronic disease.

2.5 Study Approval

The participants involved in the study were individually informed about the way of the data collection, and they did provide their consent to retrieve and process their data. The study is conformed to the General Data Protection Regulation (GDPR).

3 Method

In this section, the methodology behind the research project is discussed. Then, essential pieces of a machine learning workflow are analysed and explained step by step as follows:

- Exploratory Data Analysis
- Machine learning models
- Feature engineering for the keystroke and sensor dataset
- Feature selection per target
- Evaluation metrics
- Feature Importance

3.1 Exploratory Data Analysis

In general, classical statistics test hypotheses that already establish the problem; this is done by fitting specific models and demonstrating specific relationships in the data. It assumes that the problem hypotheses are already known, and there is an overall understanding of the data. However, this is rarely the case in machine learning. Therefore, before modelling the data and testing hypotheses, first comes the exploration of the data. Then, the relationship of understanding and exploring the data is obtained by summarizing, plotting and reviewing the actual dataset. This approach of analysis before modelling is called Exploratory Data Analysis.

It is called exploratory data analysis because the exploration and understanding of the data build an intuition for how the underlying process works and provoke questions and ideas that one can use as the basis for modelling. In addition, this process can be used to evaluate the data, identify outliers, and find specific machine learning strategies for handling any issues. [23]

Five vital findings were revealed after the exploration and understanding of the dataset. All of them are explained and argued with valuable visualizations. These five findings are explored in the following order: Missing Values, Imbalanced Targets, t-Distributed Stochastic Neighbor Embedding (t-SNE), Walking vs Rest, Chi-square statistic.

- **Missing Values:** Real-world datasets often have missing values. Data can have missing values for several reasons, such as observations that were not recorded and data corruption. Handling those missing data is critical as many machine learning algorithms do not support data with missing values, do not provide any insights and can mislead the direction of the problem. Figure 3 shows how and why the specific missing values are removed from the dataset. Features containing over 60% of missing values are discarded from the dataset, and as the figure shows, there are 29, and

the reason is that it is not feasible to make an accurate imputation of those missing values, which can yield misleading conclusions about the data and results. The shape of the new dataset after removing features that contain over 60% missing values is: (2224, 135).

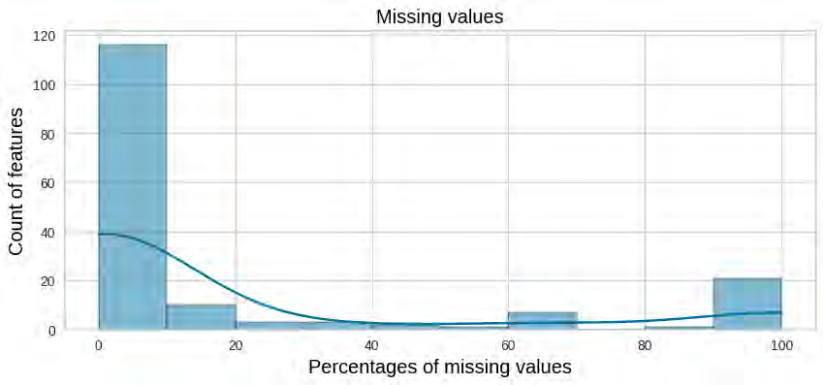


Figure 3: Missing values distribution

- **Imbalanced Targets:** In every machine learning project, the targets have to be investigated explored. Investigating the targets, in this case, three targets with multiple classes can provide insights into how classes are distributed within the targets. Furthermore, depending on the balanced or imbalanced dataset, one can use this information to select appropriate models and metrics. In this specific case, the dataset is imbalanced for all targets, and this can also be verified in Figure 4, which leads on using as metric the F1 score, which take into account imbalanced targets because it distributes the weights accordingly to all classes.

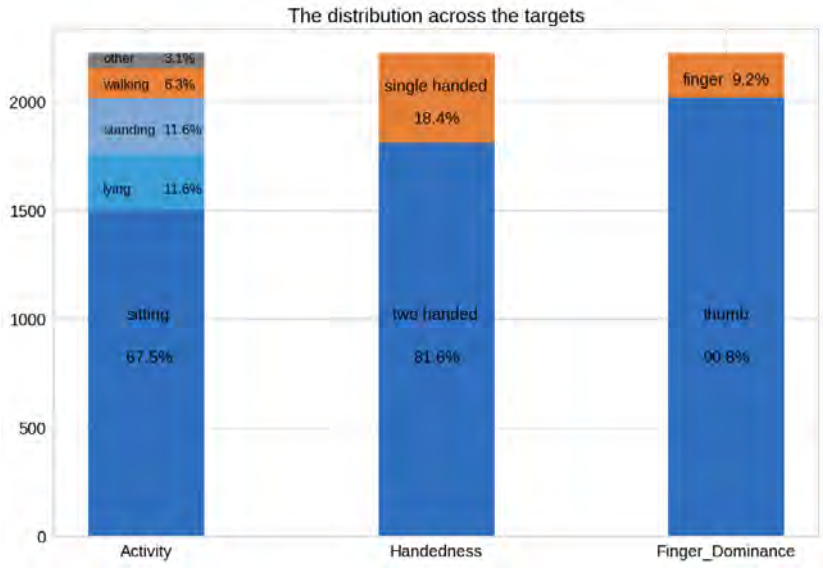


Figure 4: The distributions of the targets including the percentages for every class.

- **t-Distributed Stochastic Neighbor Embedding:** t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique used to represent a high-dimensional dataset in a low-dimensional space of two or three dimensions so that it can be visualized. First, T-SNE constructs a probability distribution for the high-dimensional samples so that similar samples have a high likelihood of being picked. Then, t-SNE defines a similar distribution for the points in the low-dimensional embedding. Finally, t-SNE minimizes the KullbackLeibler divergence between the two distributions concerning the locations of the points in the embedding.[14]

  In this case, the dimensionality of the dataset was reduced to 2 dimensions, and the KullbackLeibler divergence was 0.71. t-SNE provides valuable insights for the target activity, which as it is also shown in Figure 5 concludes that the classes "sitting", "lying", "standing", "other", do not present a clear cluster, which suggests a possible high percentage of misclassification among these classes. However, the class "walking" seems to be separated from the other classes.

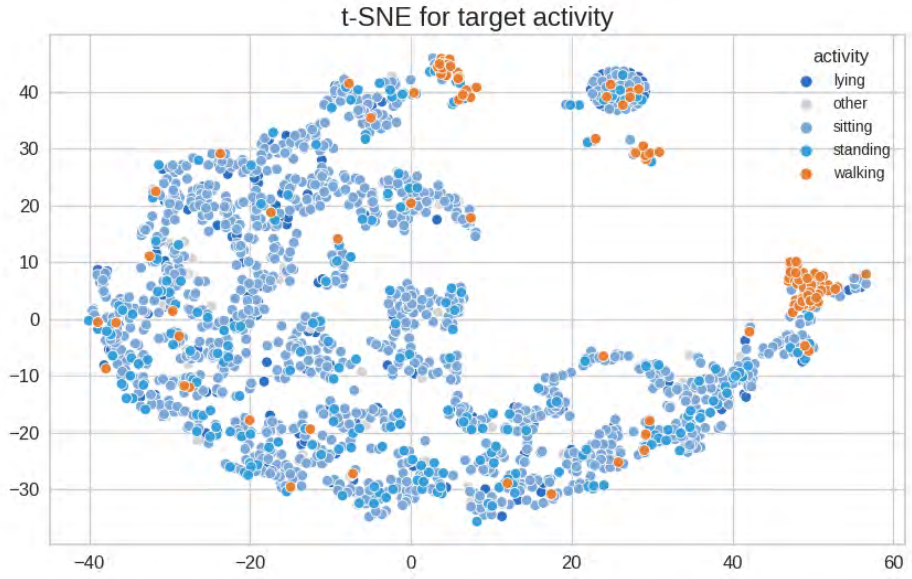


Figure 5: The dataset is visualized in 2D space using the dimensionality reduction technnique t-SNE, colored observations corresponding in different classes of the target activity

- **Walking vs Rest:** Due to the insights provided first by the t-SNE analysis, it was decided to investigate the target further "activity. This part of the exploratory analysis was critical because the transition from five classes ("walking", "sitting", "lying", "other", "standing") to two classes ("walking" vs "the rest") was decided.
  Plotting and exploring the features that are considered predictive for the class "activity" yields the conclusion that the dataset features are not capable of providing the models with information to distinguish between five classes, as four of them seem similar.
  Figure 6 shows four boxplots of the target "activity" and the features `acceleration_MAC_SAMPLE_S`, `angularACC_MOMENTS_ABV`, `acceleration_STD` and `angularVEL_STD` for the target "activity". The values for the classes "sitting", "lying", "other", "standing" are identical, thus making the case of predicting those classes difficult. The hypothesis that four classes seem to be similar is argued with these four boxplots and supports the decision to change this multiclass target into a binary target.

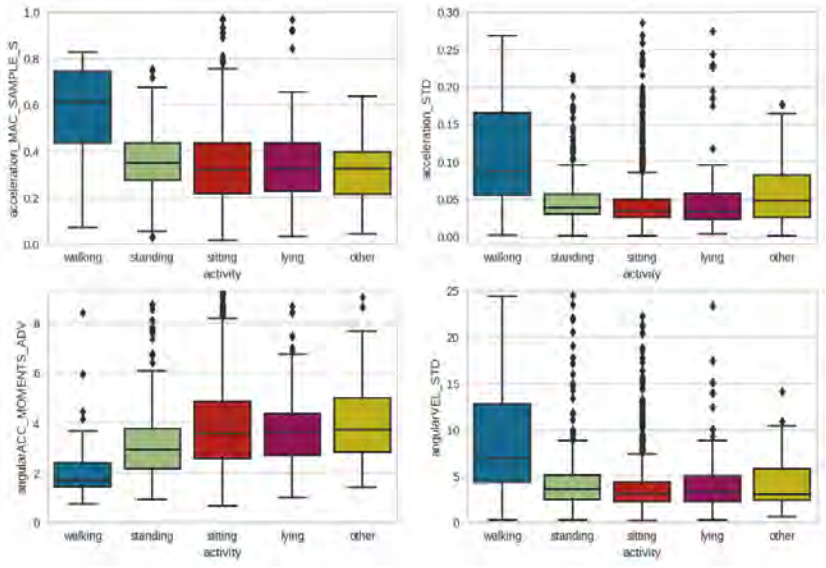


Figure 6: Walking vs Rest

In order to support the insights from the t-SNE analysis and the visualization of the features in boxplots statistically, a one-way ANOVA test is conducted to confirm that the means of these classes are significantly different. Indeed, in Table 7 the results show a significant difference between the classes of target activity for every feature mentioned above.

Table 7: ANOVA table to find if there is significant difference between the associated features

activity/features	accel_STD		accel_MACS		angularVEL_STD		angularACC_STD	
	F-stat	p-val	F-stat	p-val	F-stat	p-val	F-stat	p-val
activity-classes	28.31	2.80e-20	29.39	5.84e-21	21.11	1.55e-15	29.21	7.50e-21

Although the obtained results from the ANOVA table provided the significant difference between the classes of activity, there is no information which of them are significantly different. To find which classes have different means, one can use the pairwise Tuckey test [4]. This test conducts multiple comparisons of means in a pairwise form. Indeed, the results show that 'walking' is significantly different from the other classes, and also, the remaining classes are not significantly different between them. Hence, the decision to keep only two classes in the final dataset is confirmed statistically. Table 8 below shows the pairwise Tuckey Test.

Table 8: Pairwise comparison of Means with the Tuckey Test

Multiple Comparison of Means - Tukey HSD, FWER=0.05				
group1	group2	meandiff	p-adjust	reject
lying	other	-0.0979	0.6977	False
lying	sitting	-0.0773	0.2817	False
lying	standing	0.0489	0.8649	False
lying	walking	0.5352	0.001	True
other	sitting	0.0206	0.9	False
other	standing	0.1468	0.3341	False
other	walking	0.6331	0.001	True
sitting	standing	0.1262	0.0116	True
sitting	walking	0.6125	0.001	True
standing	walking	0.4863	0.001	True

- **Chi-square statistic [19]:** The Chi-square test is a non-parametric statistical test that enables us to understand the relationship between the categorical variables of the dataset. In addition, it defines the correlation amongst the grouping categorical data. In this case, conducting Chi-square tests among the targets (activity, handedness, finger dominance) is helpful, as it enables understanding the dependency between the targets. Similar to other statistics, Chi-square is defined by two hypotheses as follows:
  - **The null hypothesis:** The grouping variables have no association or correlation amongst them.
  - **The alternate Hypothesis:** The variables are associated with each other and happen to have a correlation between the variables.

  To summarize, the targets that are correlated based on the Chi-square tests are the following: activity and finger dominance are dependent, handedness and finger dominance are dependent, activity and handedness are independent, as it is also shown in Table 9, 10, 11 when the p-value is smaller than 0.05 the null hypothesis is rejected. As was expected, there is a strong association between the target handedness and finger_dominance because most people, for example, tend to use two hands and thumbs instead of two hands and their fingers while typing. Therefore, although the p-value between activity and handedness is close to the value 0.05, the test suggests accepting the null hypothesis that there is no association between the two targets.

Table 9: Chi-square Test between Activity and Handedness

activity/handeness	single_handed	two_handed	All
walking	35	106	141
vs Rest	375	1708	2083
ALL	410	1814	2224
Statistical Value	**p-value**	**degrees of freedom**	
4.08	0.12	2	

Table 10: Chi-square Test between Activity and Finger Dominance

activity/finger_dominance	finger	thumb	All
walking	3	138	141
vs Rest	202	1881	2083
ALL	205	2019	2224
Statistical Value	**p-value**	**degrees of freedom**	
9.04	0.01	2	

Table 11: Chi-square Test between Finger Dominance and Handedness

finger_dominance/handeness	single_handed	two_handed	All
finger	104	306	410
thumb	101	1713	1814
ALL	205	2019	2224
Statistical Value	**p-value**	**degrees of freedom**	
156.64	$9.67e$-35	2	

3.2 Machine Learning Models

This subsection is a short introduction of the models that are used to train the dataset and make the final predictions. The models are explained in the following order: general ensemble models, two specific cases of ensemble models; the random forests and the gradient boosting, multi-layer perceptron and k-neighbors. Various models are used for training; however, the top-performing models are chosen. The implementation of these models is taken only from the scikit-learn library [20] for consistency.

- **Ensemble Model:** Using the "wisdom of the crowd" baseline, ensemble models aggregate the predictions of a group of predictors (many classifiers), often yielding better results than the best individual predictor. A group of predictors is called an ensemble; thus, this technique is called ensemble learning. A simple way to create an ensemble classifier is to aggregate the predictions of each classifier and predict the class that gets the most votes. This majority-vote classifier is called a "hard" voting classifier.

  - **Random Forest:** A Random Forest is an ensemble of Decision Trees, meaning that a random forest model is made up of a large number of small decision trees, called estimators, which produce their predictions, generally trained via the bagging method. The Random Forest introduces extra randomness when growing trees; instead of searching for the very best feature when splitting a node, it searches for the best feature among a random subset of features. As a result, standard decision tree classifiers have the disadvantage of overfitting the training set. The random forest's ensemble design allows the random forest to compensate for this and generalize well to unseen data, including data with missing values [2].
  - **Gradient Boosting:** Boosting refers to any ensemble method combining several weak learners into a strong learner. The idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor. For example, gradient Boosting [5] instead of tweaking the instance weights at every iteration; tries to fit the new predictor to the residual errors made by the previous predictor.

- **Multi Layer Perceptron:** A multi-layer perceptron is a neural network connecting multiple layers in a directed graph, which means that the signal path through the nodes only goes one way. Apart from the input nodes, each node has a nonlinear activation function. This specific network uses a technique called backpropagation as a supervised learning technique. Since there are multiple layers of neurons, the multi-layer perceptron is considered a deep learning technique [8].

- **K-Neighbors:** K-Nearest Neighbors (KNN) is a classification and algorithm which uses nearby points to generate predictions. It takes a point, finds the K-nearest points, and predicts a label for that point, K being user-defined. For classification, the algorithm uses the most frequent class of the neighbors. KNN, being a distance-based classifier, can use different types of distance metrics to calculate similarity, such as the Euclidean distance, Manhattan distance, and Minkowski distance. Being a non-parametric algorithm, it does not have coefficients [21].

3.3 Feature Engineering

According to [18], feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modelling. The goal of feature engineering and selection is to improve the performance of machine learning (ML) algorithms. Feature engineering consists of creating, transforming, extracting, and selecting features, also known as variables, that are most conducive to creating an accurate machine learning algorithm.

- **Feature Creation:** Creating features involves identifying the variables that will be most useful in the predictive model; this is a subjective process

that requires human intervention and creativity. Then, existing features are mixed via addition, subtraction, multiplication, and ratio to create new derived features with greater predictive power.

- **Transformations:** Transformation involves manipulating the predictor variables to improve model performance; e.g. ensuring the model is flexible in the variety of data it can ingest; ensuring variables are on the same scale, making the model easier to understand; improving accuracy; and avoiding computational errors by ensuring all features are within an acceptable range for the model.
- **Feature Extraction:** Feature extraction automatically creates new variables by extracting them from raw data. The purpose of this step is to automatically reduce the volume of data into a more manageable set for modelling. Some feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis.

In order to achieve the goals of feature engineering and keep only the most valuable features of the dataset, the composite scores are introduced to reduce the number of features and maintain the information of the previous features. According to [1], the composite scores represent small sets of data points that are highly related to one another, both conceptually and statistically. Thus, combining and presenting these items as a single score reduces the potential for information overload.

3.3.1 Keystroke Dataset

The concept of composite scores is applied to the keystroke dataset. Table 12 shows an example of the way some features are aggregated. Using the composite scores, the features are reduced from 388 to 84 and most of the information is maintained. Both features presented in Table 12 are time-related features and measure how fast or slow a person is typing.
In this work, `fine_motor_score_CENTRALITY` is considered a feature with high predictive power because it considers the general trend of a given typing session, which can lead to some expectations on common sense. People while walking and typing are expected to be slower than when sitting, or similar, when typing just with one hand instead of two. Also, it is common to type with the thumb while walking because it is more convenient. All these expectations are time-related and expressed based on the `fine_motor_score_CENTRALITY` values. In general, high values of this feature means slower typing.

Table 12: The clustering some of the keystroke features

fine_motor_score_CENTRALITY	**fine_motor_score_STD**
flight_time_MEAN	flight_time_STD
press_press_latency_MEAN	press_press_latency_STD
release_release_latency_MEAN	release_release_latency_STD
flight_time_MEDIAN	-
press_press_latency_MEDIAN	-
release_release_latency_MEDIAN	-

3.3.2 Sensor Dataset

As discussed in Subsection 2.2.3, the sensor dataset is originated from the gyroscope and the accelerometer dataset. Both datasets contain many features, specifically the gyroscope dataset consists of 268 and the accelerometer dataset consists of 94 features. For the same purpose as for the keystroke dataset, reducing the number of features is required.
For each feature of the sensor data (rotation, angular velocity, angular acceleration for the gyroscope and acceleration features for the accelerometer), five summary statistics (i.e. vectors) are calculated: mean and median (indicators of central tendency), standard deviation (SD; an indicator of dispersion) and minimum and maximum (indicators of range). In addition, other statistics are also calculated and added as vectors of the feature as skew, kurtosis, the last and first max value, the mean absolute change and the sample entropy. Definitions of those statistics are as follows:

- **Mean Absolute Change:** Average of the absolute change (positive) between a point in a sequence and the next point.
- **Sample Entropy:** Measure of time series complexity. Is the negative natural logarithm between two template vectors [22].

In order to reduce the size of the features, correlations and distributions of the features are investigated. High correlated features with identical distributions are averaged to reduce the features but keep the overall information of the data and features. The averaging for the gyroscope features is performed based on the same unit features, meaning that rotations (pitch, yaw and roll) are investigated together, angular velocity (x, y, z) are investigated together, and the same applies to angular acceleration (x, y, z) features. The averaging of the accelerometer features is produced as for the acceleration (x, y, z) of the gyroscope. The initial feature size of the accelerometer and the gyroscope are 57 and 267. After performing the composite score analysis and aggregating the features into new information, the feature size of the accelerometer and gyroscope datasets are 21 and 93, respectively. Understandably, the reduction of the features is substantial without removing much information power from that dataset. Table 13 shows an example of the way some features are aggregated.

The correlation of those features is high, as shown in Figure 7, thus a composite score is constructed.

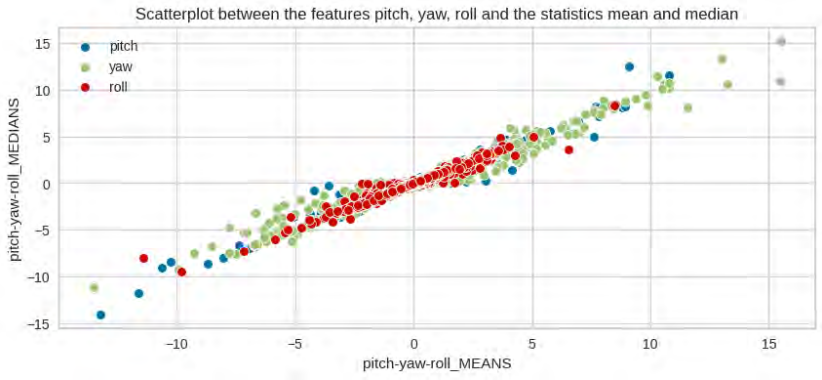


Figure 7: Scatterplot of pitch, yaw and roll features, originating from the gyroscope that express the rotational degrees of the smartphone and the aggregations of the mean and the median in the given typing session. The data points are linear with high correlation.

Table 13: The clustering of sensor features

rotational_CENTRALITY	**angularVEL_CENTRALITY**	**angularACC_CENTRALITY**
pitch_MEAN	omega_x_MEAN	x_filtered_MEAN
yaw_MEAN	omega_y_MEAN	y_filtered_MEAN
roll_MEAN	omega_z_MEAN	z_filtered_MEAN
pitch_MEDIAN	omega_x_MEDIAN	x_filtered_MEDIAN
yaw_MEDIAN	omega_y_MEDIAN	y_filtered_MEDIAN
roll_MEDIAN	omega_z_MEDIAN	z_filtered_MEDIAN

3.4 Feature Selection

According to [10], feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modelling and, in some cases, improve the model's performance. Statistical-based feature selection methods involve evaluating the relationship between each input variable and the target variable using statistics and selecting those input variables that have the most robust relationship with the target variable. These methods can be fast and effective, although statistical measures depend on the data type of both the input and output variables.

In order one to find the best input features, creates many models with different subsets, which results in the best performing model based on the chosen performance metric, which in this work is the F1 score and is explained in subsection 3.6; this approach is called wrapper feature selection. Although these methods can be computationally expensive, these methods are unconcerned

with the variable types. Recursive feature elimination (RFE) [7] is an example of a wrapper feature selection method and is also the technique that is used in order to select the best features for every target separately. This wrapper method evaluates multiple models using procedures that add and/or remove predictors to find the optimal combination that maximizes model performance. Cross-validation is used with RFE to find the optimal number of features, score different feature subsets and select the best scoring collection per model.

The cross-validation scheme that is used for the feature selection part is the Stratified K-Fold [20]. This cross-validation scheme provides train/test indices to split data into train/test sets. This cross-validation object is a simple KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class and this is why this scheme is selected.

Four models are used to complete the RFE with cross-validation, decision trees, random forest, gradient boosting model and AdaBoost, which are explained in Section 3.2. The metric that is used to evaluate the performance is the F1 score, which is explained in Section 3.6. The small dataset is constructed from the union of features that the models produce, which is used later for training and testing. The union of the recommended features is preferred because this methodology increases the robustness as every model has its mathematical structure, and if all of them agree on the same feature, then there is the confidence that those features contain predictive power.

In order to perform RFE with cross-validation for each target separately, the target is separated from the dataset. The targets are considered as binary targets as they include two classes. The models learn from the combination of features of both datasets, the keystroke and sensor dataset. Figures 8, 9, 11 show the performance of each model and for every target. In the x-axis are located the number of features used within the cross-validation scheme, and on the y-axis is the F1 score, achieved every time the model removed a feature. Furthermore, Figures 8, 9, 11 show the results after training and evaluating, where for every model, the best F1 score is kept along with the number of features used.

While the F1 scores of the models are not far away from the best score, and because Figures 8, 9, 11 show that the best model selects a high number of features, thus it is optimal to apply the union of the features produced by the models to construct the mini datasets for every target, in order to create a robust dataset with predictive power.

Figure 8 shows that the random forest model performs the best by scoring 0.95 and using 17 features. Applying the union to construct the mini activity dataset, which finally, contains the following features: `angularVEL_STD`, `angularACC_MOMENTS_ADV`, `acceleration_MAC_SAMPLE_S`. It was expected that the best features would come from the sensor dataset because the target activity implies an underlying movement either from the subject or the smartphone.

Figure 8: RFE with cross validation for target activity. The random forest classifier achieves the highest score at 0.954 and selects 17 features.

In Figure 9, AdaBoost performs the best by scoring 0.828 and using 19 features. Applying the union to construct the mini handedness dataset, which finally, contains the following features: `fine_motor_score_CENTRALITY`, `fine_motor_score_MIN`. Target handedness implies that the subject will be evaluated based on how quick and accurate is typing with one or two hands, and this is the reason why features that measure the motor skills are selected as predictive features for this target, which was expected, due to the nature of this target.

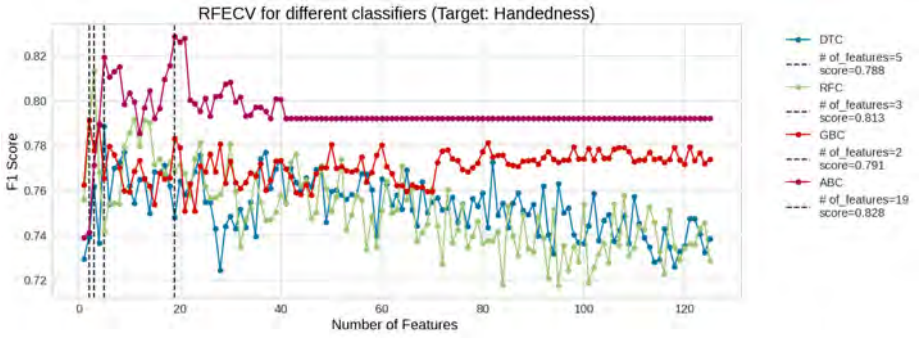


Figure 9: RFE with cross validation for target handedness. The AdaBoost classifier scores 0.828 by selecting 19 features.

To support the selection, Figure 10 shows indeed that the choice of those features can yield promising results on the final evaluation. The minimum values for the fine motor score class are lower in the case, which the subject is typing with two hands than typing with one hand. The participants are responding quicker, either when it comes to fast typing or correcting a mistake when typing with two hands.

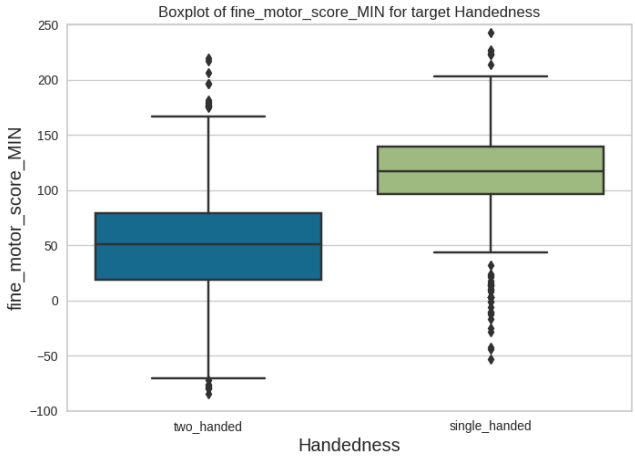


Figure 10: The boxplot of the *fine_motor_score_MIN* feature, which is the average of the minimum values of the features *flight_time*, *press_press_latency* and *release_release_latency* in a given typing session, one can observe that the fastest keypress recorded using two hands is roughly twice times faster than the scenario with one hand.

Figure 11 displays the Gradient Boosting classifier, which performs the best by scoring 0.839 and using seven features. Applying the union to construct the mini finger dominance dataset, which finally, contains the following features:: `acceleration_STD`, `angularACC_BOUNDS_MAX`, `fine_motor_score_CENTRALITY`. This target implies that the participants are typing either with their thumbs or index fingers. As many people are performing their typing sessions using the thumbs, because it is more convenient than typing with the index finger, it is expected that at least one feature would have been chosen from the keystroke dataset, as typing with the thumb practically means typing faster and more accurate. Also, there is an association between the targets finger dominance and activity, as it is shown in Section 3.1 by performing the chi-square test. Because it is expected the users to use their thumb while walking, the models use features associated with the activity target to understand the possibility of a person walking in the specific typing session and then associate this possibility with the motor skills measures to predict either thumb or finger.

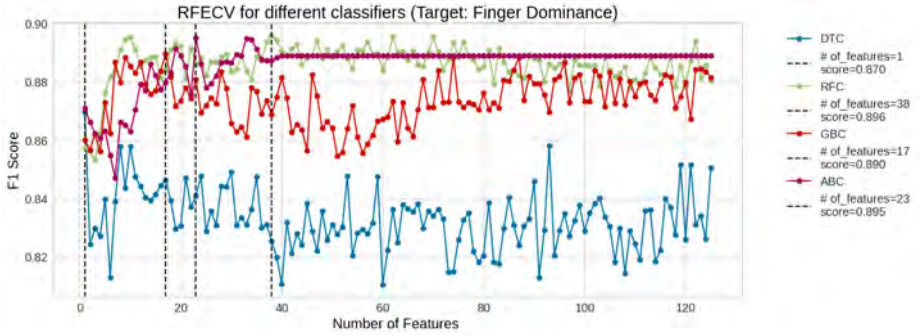


Figure 11: RFE with cross validation for target finger dominance

3.5 Data Preparation & Machine Learning Approaches

In this subsection, the process is explained, which the models are prepared to be trained, which is similar for both approaches that are used to predict the boundary conditions and the approaches themselves. In machine learning, preparing the data is an integral part of a predictive modelling process.

Correct data preparation application will transform raw data into a representation that allows learning algorithms to get the most out of the data and make skilful predictions. It is also common practice to evaluate machine learning models on a dataset using k-fold cross-validation scheme.

The custom pipeline used for preparing the dataset includes the following steps explained in detail below: Imputation of missing values, detecting outliers, scaling the data for the models that need scaled data, and the models' hyperparameters.

- **Imputation** Datasets may have missing values, and these cause problems for many machine learning algorithms. Thus, it is vital to identify and replace missing values for each column in the training data before modelling any prediction task. This is called missing data imputation or imputing for short. A popular approach to missing data imputation is to use a model to predict the missing values. This requires a model to be created for each input variable that has missing values. Although a range of different models can be used to predict the missing values, the k-nearest neighbor (KNN) algorithm has proven to be effective [20], often referred to as "nearest neighbor imputation". Configuration of KNN imputation often involves selecting the distance measure (e.g. Euclidean) and the number of contributing neighbors for each prediction, the k hyperparameter of the KNN algorithm. Each sample's missing values are imputed using the mean value from "k_neighbors" nearest neighbors found in the training set. Two samples are close if the features that neither is missing are close.

- **Outlier detection** A dataset can contain extreme values outside the

range of what is expected and unlike the other data. These are called outliers, and often machine learning modelling can be improved by understanding and even removing these outlier values. An outlier is an observation that is unlike the other observations. It is rare, or distinct, or does not fit somehow. Unfortunately, there is no precise way to define and identify outliers in general because of the specifics of each dataset.

The chosen method of identifying and removing the outliers of the dataset is the interquantile range method (IQR). In this work, the interquantile range is calculated as the difference between the 95th and the 5th percentiles of the data. The IQR can be used to identify outliers by defining limits on the sample values that are a factor k of the IQR below the 5th percentile or above the 95th percentile. The standard value for the factor k is the value 1.5. A factor k of 3 or more can be used to identify values that are extreme outliers or "far outs" when described in the context of box and whisker plots. Figure 12 shows how the dataset is formed before and after applying the IQR method in different percentiles. Figure 13 shows an example of outlier removal.

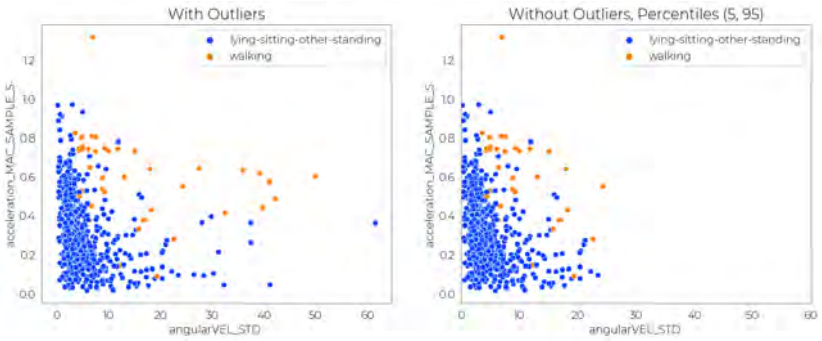


Figure 12: Example of two features prior and post outlier removal operation with focus on activity target. In this specific case the method discards all sample of *angular_vel_STD* greater than 25 rad/s.

- **Robust Scaling** The method that is used to scale the data is Robust scaling [20]. This method scales features using statistics that are robust to outliers. This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range). The IQR ranges between the first quartile (25th quantile) and the third quartile (75th quantile). Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set.

- **Model Tuning** The last step of the custom pipeline for preprocessing the dataset before fitting it into a training model is to specify the model and the hyperparameters. The hyperparameters are fine-tuned through the

process of grid searching, that is, searching in a specific set of parameters the ones that fit the dataset the best. Tables 14, 15, 16 shows exactly which hyperparameters are tuned for each model and the specific values chosen for each model. Fine-tuning is performed for every binary target separately.

Table 14: The hyperparameters and the models for target Activity

Activity Gridsearch Parameters					
Gradient Boosting		K-Neighbors		Random Forests	
Param Name	Param Value	Param Name	Param Value	Param Name	Param Value
n_estimators	200	algorithm	brute	criterion	entropy
learning_rate	0.1	leaf_size	10	max_depth	10
loss	exponential	n_neighbors	8	max_features	auto
min_samples_leaf	4	weights	uniform	class_weights	balanced
subsample	0.9			n_estimators	200
max_depth	4				

Table 15: The hyperparameters and the models for target Handedness

Handedness Gridsearch Parameters					
Gradient Boosting		Random Forests		Multi Layer Perceptron	
Param Name	Param Value	Param Name	Param Value	Param Name	Param Value
n_estimators	150	criterion	entropy	hidlayer_sizes	100
learning_rate	0.1	max_depth	10	activation	relu
loss	exponential	max_features	auto	alpha	0.0001
min_samples_leaf	3	class_weights	balanced	learning_rate	constant
subsample	0.9	n_estimators	200	max_iter	300
max_depth	4				

Table 16: The hyperparameters and the models for target Finger Dominance

Finger Dominance Gridsearch Parameters					
Gradient Boosting		K-Neighbors		Multi Layer Perceptron	
Param Name	Param Value	Param Name	Param Value	Param Name	Param Value
n_estimators	150	algorithm	brute	hidlayer_sizes	100
learning_rate	0.1	leaf_size	10	activation	relu
loss	exponential	n_neighbors	8	alpha	0.0001
min_samples_leaf	3	weights	uniform	learning_rate	constant
subsample	0.9			max_iter	300
max_depth	4				

3.5.1 Binary Target Classification

The first approach is binary classification. Each target is distinguished into two classes, and for each target, different models are trained separately with different features taken from the original dataset. The targets and the classes are shown in Table 17.

Table 17: The targets and the classes

Targets	Classes
Activity	Walking vs Non-motion
Finger Dominance	Finger vs Thumb
Handedness	Single vs Two Handed

3.5.2 Multilabel Classification

The second approach is the multilabel classification, where the target is considered multilabel because the models are trying to distinguish the boundary conditions of the three targets simultaneously. All the features used to predict each target separately are considered in the training of the multilabel classification, in total, eight features. Table 18 is the nature of the features that are used to investigate the boundary conditions of each target and of the multilabel classification.

Table 18: The targets and the features

Targets	Features
Activity	Only 3 sensor features
Finger Dominance	Mix of 1 sensor and 2 keystroke features
Handedness	Only 2 keystroke features

3.6 Evaluation Metrics

Choosing an appropriate metric is generally challenging in applied machine learning but is particularly difficult for imbalanced classification problems. Firstly, because most of the standard metrics widely used assume a balanced class distribution, and because typically not all classes, and therefore, not all prediction errors, are equal for imbalanced classification.

An evaluation metric quantifies the performance of a predictive model. It typically involves training a model on a dataset, using it to make predictions on a holdout dataset not used during training, then comparing the predictions to the expected values in the holdout dataset. For example, for classification problems, metrics involve comparing the expected class label to the predicted class label or interpreting the predicted probabilities for the class labels for the problem.

Importantly, different evaluation metrics are often required when working with imbalanced classification. For example, unlike standard evaluation metrics that treat all classes as equally important, imbalanced classification problems typically rate classification errors with the minority class as more important than those with the majority class. As such, performance metrics may be needed that focus on the minority class, which is made challenging because it is the minority class where we lack the observations required to train an effective model.

Important terminology needed to understand the following metrics:

- **True Positives or TP:** Are the test samples that are correctly predicted from the class considered as the positive class.
- **False negatives or FN:** Are the test samples that are wrongly predicted from the class considered as the positive class.
- **True Negatives or TN:** Are the test samples that are correctly predicted from the class considered as the negative class.
- **False positives or FP:** Are the test samples that are wrongly predicted from the class considered as the negative class.

Selecting a model and even the data preparation methods together is a search problem guided by the evaluation metric. Experiments are performed with different models, and the outcome of each experiment is quantified with a metric. These metrics used to quantify the performance of the models are the following:

- **F1-score:** Precision and recall can be combined into a single score that seeks to balance both concerns, called the F-score or the F1-measure. The F1-score is a popular metric for imbalanced classification. It is considered as the harmonic mean of precision and recall. Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values.

$$\text{F1-score} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

- **Precision:** Precision summarizes the fraction of examples assigned the positive class that belong to the positive class. In other words is the accuracy of the positive predictions.

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **Recall:** Recall summarizes how well the positive class was predicted and is the same calculation as sensitivity. It is also called sensitivity or the true positive rate (TPR), which is the ratio of positive instances that are correctly detected by the classifier.

$$\text{Recall} = \frac{TP}{TP+FN}$$

- **Confusion Matrix:** The confusion matrix is a summary of prediction results on a classification problem. The correct and incorrect predictions are summarized with count values and broken down by each class. It is the key to the confusion matrix. The confusion matrix shows how any classification model is confused when making predictions. It gives insight into the errors being made by the classifier and, more importantly, the types of errors that are being made. An instance of a confusion matrix can be seen in Figure 13.

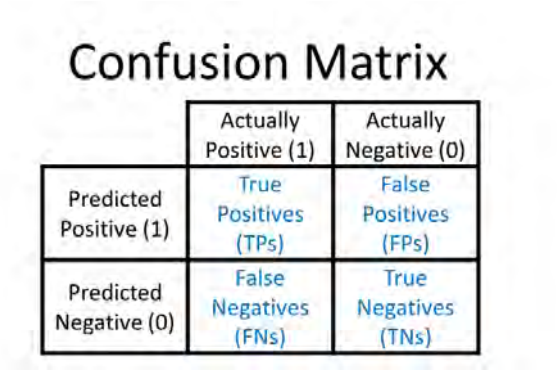


Figure 13: Example confusion matrix, [6]

- **ROC - AUC score:** ROC is an acronym that means Receiver Operating Characteristic and summarizes a field of study for analyzing binary classifiers based on their ability to discriminate classes. A ROC curve is a diagnostic plot for summarizing the behaviour of a model by calculating the false positive rate and true positive rate for a set of predictions by the model under different thresholds. Each threshold is a point on the plot, and the points are connected to form a curve. A classifier with no skill (e.g. predicts the majority class under all thresholds) will be represented by a diagonal line from the bottom left to the top right. Any points below this line have worse than no skill. A perfect model will be a point in the top left of the plot. The ROC Curve is a helpful diagnostic for one model. The area under the ROC curve can be calculated and provides a single score to summarize the plot used to compare models. A no skill classifier will have a score of 0.5, whereas a perfect classifier will have a score of 1.0. The true positive rate is the recall or sensitivity. Hence, the ROC curve plots sensitivity (recall) versus 1 - $specificity$.

3.7 Feature Importance with SHAP

Machine learning applications need to understand why the models are making confident predictions. However, this part is often difficult because the interpretation of the model's behaviour is not straightforward due to the underlying naive tension between the performance measures and the interpretability of the models.

A unified framework for interpreting predictions is used to address this problem, namely, SHapley Additive exPlanations (SHAP). As it is described in [13] SHAP assigns each feature an importance value for a particular prediction. Its novel components include (1) identifying a new class of additive feature importance measures and (2) theoretical results showing a unique solution in this class with a set of desirable properties.

Each point is a Shapley value for an instance and a feature. The position on the x-axis is determined by the Shapley value and on the y-axis by the feature. Thus, the blue color represents a lower feature value, whereas red represents a higher value. The features are ordered according to their importance, and the summary plot replaces the typical bar chart of feature importance. It explains which features are most important and their range effects over the dataset. The process of investigating the importance of the features per target is shown below. Note that when points do not fit together on the line, they pile up vertically to show density. Each dot is also colored by the value of the feature from high to low.

4 Results

In this section, the results will be described. This section is organized as follows: three primary levels are introduced for every target separately. Firstly, presenting the ROC Curves of the hold-out set, the confusion matrices, and the feature importance per target.

4.1 Activity

This target aimed to investigate if the person who is typing is walking or performing any other activity, and the model achieved the goal with a very high degree of certainty. Table 19 shows the obtained results after evaluating the model on the final Hold-out set. The F1 score (0.927), which is the desired performance metric, is considered more than acceptable.

Table 19: The activity results

Target/Metrics	F1 score	Precision	Recall	Accuracy	AUC Score
Activity (walking vs rest)	0.927	0.932	0.932	0.932	0.92

In Figure 14 it is shown the Receiver Operating Characteristic curve, both for the training and the Hold-out set. Unexpected, the model performs better in the case of the Hold-out set, which is due to a combination the concept of data shifting [15] issue as well as the way how the train and test set were splitter. The blue line represents the average value of the AUC score, for the training set after five validation folds, with an AUC score of 0.77. The yellow line represents the AUC score on the Hold-out set, which yielded an AUC score of 0.92, and the red dashed line represents a model that has no skill in predicting the target activity.

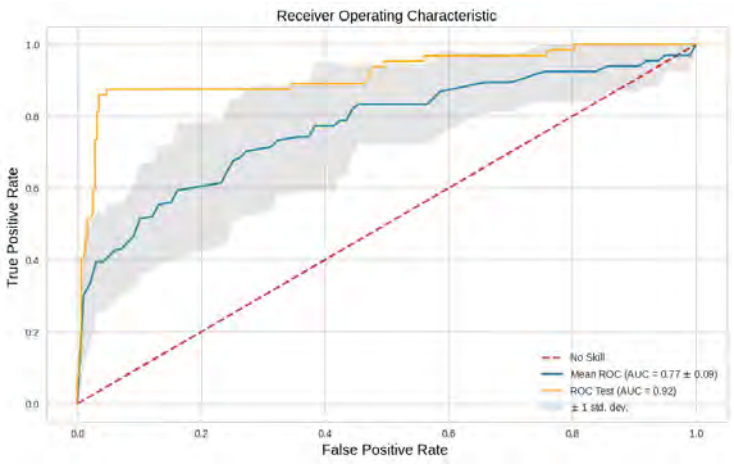


Figure 14: ROC for target Activity

Figure 14 shows unexpected performance on the hold-out set. The AUC score on the hold-out set is considerably greater than on the training set; however, this can be explained by the concept of data shifting. Figure 15 visualizes precisely the concept of data shifting, where on the training set there are different data points, hence the model is trained on those situations. There are no observations on the hold-out set for the feature `acceleration_MAC_SAMPLE_S` greater than the value of 0.75.

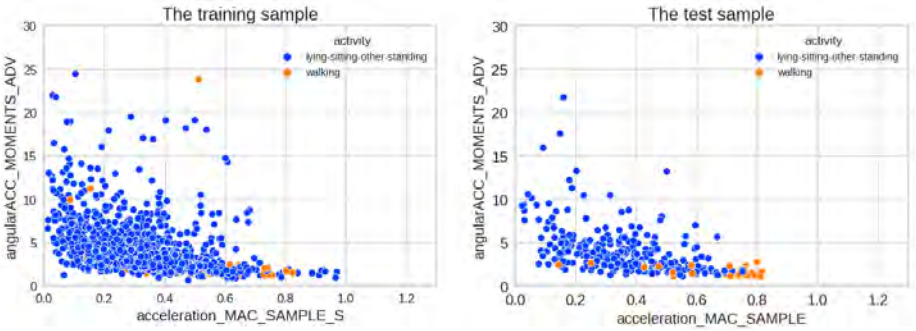


Figure 15: Data shifting target Activity

When explicitly investigating the mistakes for target activity, it is revealed that the final model scores might be optimistic. Part of this information is available in the confusion matrix, which allows checking the mistakes the model made per class like described in Subsection 3.6. For example, figure 16 shows precisely the number of misclassified "walking" typing sessions that the model predicts and the ratio computed per class. The model predicts correctly 45 typing sessions out of 64, which were included in the Hold-out set. However, the model is very confident in predicting the class "sitting-lying-standing-other" with a correct predicted typing session ratio of 0.972.

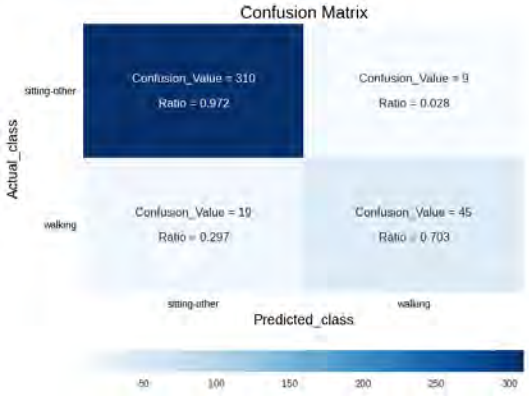


Figure 16: Confusion Matrix for target Activity

A SHAP value for a feature of a specific prediction represents how much the model prediction changes when the feature is observed. All the SHAP values are plotted in the activity summary plot for a single feature (such as `angularVEL_STD`) on a row, where the x-axis is the SHAP value. By doing this for all features, one can see which features drive the model's prediction, such as `angularVEL_STD`, which is the main contributor and only affects the predictions on a little (such as `angularACC_MOMENTS_ADV`). Specifically, higher values for the `angularVEL_STD` and the `acceleration_MAC_SAMPLE_S` features have a high and positive impact on the prediction of the class "walking", and that is the reason as people that are not walking while typing tend to have low or none velocity or acceleration; however, the feature `angularACC_MOMENTS_ADV` is negatively correlated with the class "walking".

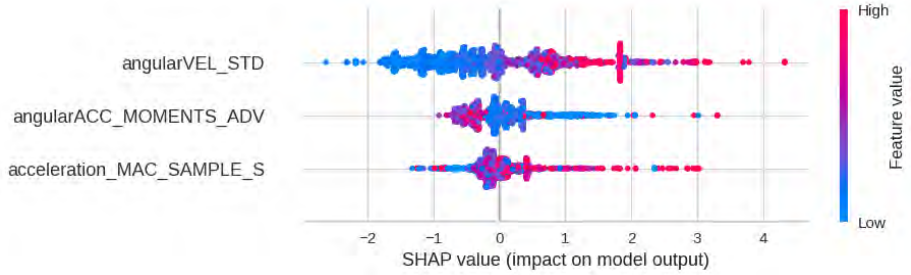


Figure 17: Shap feature importance plot for target Activity

4.2 Handedness

This target aimed to investigate if the person who is typing is using one or two hands, and the model achieved the goal with a very high degree of certainty. Table 20 shows the obtained results after evaluating the model on the final Hold-out set. The F1 score (0.818), the desired performance metric, is considered more than acceptable.

Table 20: The handedness results

Target/Metrics	F1 score	Precision	Recall	Accuracy	AUC Score
Handedness	0.818	0.844	0.803	0.803	0.84

Again for this target, before evaluating the results of the Hold-out set, the training set (only two users) was evaluated through the cross-validation scheme. In Figure 18 it is shown the Receiver Operating Characteristic curve, both for the training and the Hold-out set. The model is performing slightly better in the case of the Hold-out set. The blue line represents the average value of the AUC score, which is at 0.81, for the training set after five validation folds. The yellow line represents the AUC score on the Hold-out set with an AUC score of

0.84, and the red dashed line represents a model that has no skill in predicting the target activity.

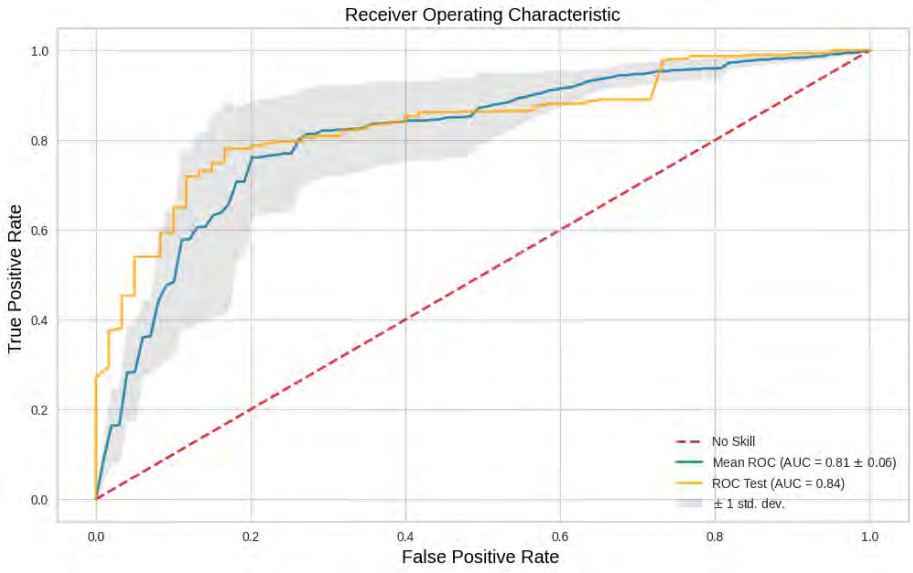


Figure 18: ROC for target Handedness

Also, in the case of the "Handedness" target, the confusion matrix allows identifying the incorrectly misclassified typing sessions that the model predicted as "single_handed". Indeed, the model is very optimistic as the ratio of the correct classified "single_handed" typing sessions is 0.60. However, the model is very confident in predicting the class "two_handed", with a ratio for the correct predicted typing sessions at 0.844.

Figure 19: Confusion Matrix for target Handedness

In the handedness summary plot, all the SHAP values are plotted for a single feature (`fine_motor_score_MIN`) on a row, where the x-axis is the SHAP value. Doing this for all features allows one to see which features drive the model's prediction, such as `fine_motor_score_MIN`. Specifically, higher values for the `fine_motor_score_MIN` and the `fine_motor_score_CENTRALITY` features have a high and negative impact on the prediction of the class "single_handed", which can also be verified if one breaks down the `fine_motor_score` class. The `fine_motor_score` class contains information about the flight times and the press and releases of the keystrokes, which explains the higher impact of lower values in the "single_handed class", as people are often typing faster when they are using two hands.

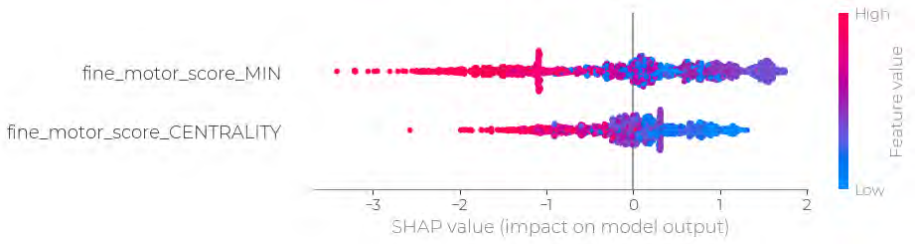


Figure 20: Shap feature importance plot for target Handedness

4.3 Finger Dominance

This target aimed to investigate if the person who is typing is using the thumb or the finger, and the model achieved the goal with a very high degree of certainty. Table 21 shows the obtained results after evaluating the model on the final Hold-out set. The F1 score (0.909), the desired performance metric, is considered more than acceptable.

Table 21: The finger dominance results

Target/Metrics	F1 score	Precision	Recall	Accuracy	AUC Score
Finger Dominance	0.909	0.913	0.924	0.924	0.80

In Figure 21 it is shown the Receiver Operating Characteristic curve, both for the training and the Hold-out set. As expected, the model performs better in the case of the training set. The blue line represents the average value of the AUC score at 0.89, as described in subsection 3.6, for the training set after five validation folds. The yellow line represents the AUC score on the Hold-out set (0.80), and the red dashed line represents a model with no skill in predicting the target activity.

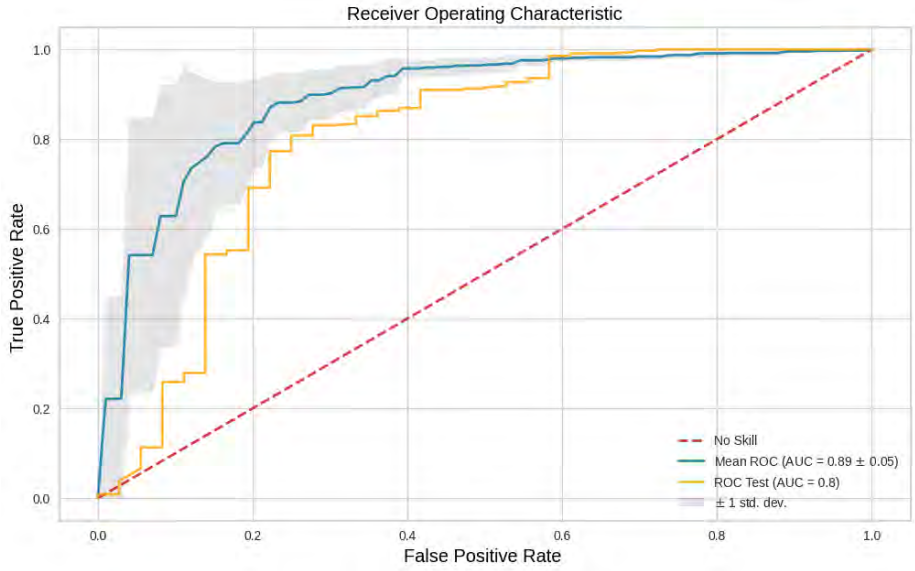


Figure 21: ROC for target Finger Dominance

In the case of the "Finger_Dominance" target, the confusion matrix allows identifying the incorrectly misclassified typing sessions that the model predicted as "finger", which seem overall very promising. However, the model is optimistic as the ratio of the correct classified "finger" typing sessions are at a ratio of 0.417. However, the model is very confident in predicting the class "thumb", with a ratio of 0.985. Only 36 typing sessions are provided on the Hold-out set; on the contrary, the typing sessions labelled as "thumb" are 343.

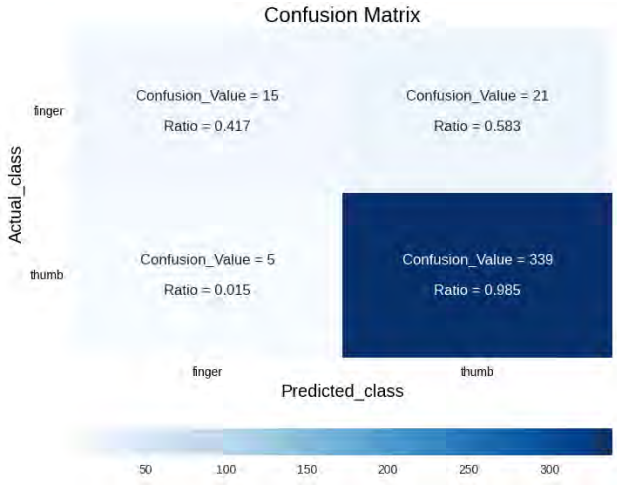


Figure 22: Confusion Matrix for target Finger Dominance

In the finger dominance summary plot, all the SHAP values are plotted for a single feature (such as `fine_motor_score_CENTRALITY`) on a row, where the x-axis is the SHAP value. Specifically, lower values for the `acceleration_STD` feature have a high and negative impact on predicting the class "finger", and higher values for the `fine_motor_score_CENTRALITY` feature have medium to high and positive impact on the prediction class "finger". Depending on the values of the feature `acceleration_STD`, any typing session can be described as active or not. For example, while a person is typing with the finger, a requirement is a motionless position for the smartphone (especially, in case of typing with both hands and fingers, then the smartphone needs to be placed on an object, so 0 acceleration). Hence, lower values of the feature `acceleration_STD` are highly associated with the class "finger".

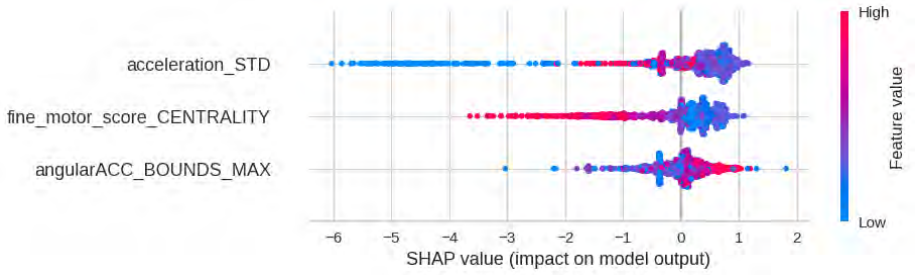


Figure 23: Shap feature importance plot for target Finger Dominance

Table 22 shows the summary results of the whole project. The most accurate predicted target is Activity, where the model predicts with F1 score at 0.927, which is the best F1 score amongst the targets and is the performance metric in which the models are optimized.

Table 22: The summary results

Targets/Metrics	F1 score	Precision	Recall	Accuracy	AUC Score	Exact Match Ratio
Activity (walking vs rest)	0.927	0.932	0.932	0.932	0.92	-
Finger Dominance	0.909	0.913	0.924	0.924	0.80	-
Handedness	0.818	0.844	0.803	0.803	0.84	-
Multilabel	0.908	0.908	0.911	-	-	0.705

5 Discussion

From Section 4 the output and predictions of the models are considered as promising for every target. However, there is still room for improvement, which is addressed in this Section.

The classification problem that is tackled in this research project is to identify the boundary conditions given a typing session; by boundary conditions, it is meant to investigate if the person is typing with or two hands if the person is typing with the thumb or finger and the activity the person is performing. Therefore, it is understandable that the project has limitations and considers some assumptions that influence the final result.

Healthy people conduct the research and data collection, so the models are trained and evaluated on data from healthy people. In order to have a clear outcome, the models have to be tested on patients and evaluate the results. If the models perform accurately also on patients data, then it can be concluded that patients behaviour is not far from healthy people, at least in investigating the boundary conditions. Furthermore, it is assumed that people who participated in the research and data collection have labelled their typing sessions honestly, which is an important parameter that can affect the outcome of the project. Also, the size of the dataset is considered decent, but the number of the participants is small. The same project with more data points, balanced classes and more participants could yield in totally different results.

A vital aspect to mention is that all three targets have imbalanced classes. In order to tackle this problem in this research problem, the F1 score is used as the primary evaluation metric because it takes into account imbalanced classes and is explained in Subsection 3.6. However, other techniques can be used to tackle the problem of imbalanced classes. Techniques like oversampling the minority class or undersampling the majority can be proved to be valuable to increase the performance of the models.

6 Conclusion

In conclusion, the research project has produced encouraging results for every target that can be further improved. In this project, two approaches have been addressed, the binary target classification and the multilabel classification. The binary target classification approach includes the following classes: for target activity; walking vs rest, for target handedness; single handed vs two handed and for target finger dominance; finger vs thumb. The multilabel classification approach includes all the classes simultaneously, and a prediction is considered correct only when the model predicts all three classes correct of a typing session.

The research project established the following research question: "Can a machine learning approach investigate the boundary conditions of a typing session?", the question is answered and argued in the previous Sections and with a high degree of certainty that the boundary conditions can be predicted, although, the models are not very confident in predicting the minority class for the target finger dominance.

As a final takeaway message, the boundary conditions of a given typing session can be predicted with a high degree of certainty based on the features produced by the Neurokeys app and is recommended further research taking into account the above limitations and assumptions that have been already discussed in Section 5.

References

[1] Agency for Healthcare Research and Quality, Rockville, MD. *Combining Healthcare Quality Measures Into Composites or Summary Scores*. [Online; accessed 30-June-2021]. URL: https://www.ahrq.gov/talkingquality/translate/scores/combine-measures.html.

[2] Leo Breiman. "Random Forests". In: *Machine Learning* 45.Oct (2001), pp. 1573–0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[3] Paul Dagum. "Digital biomarkers of cognitive function". In: *npj Digital Medicine* 1 (2018). ISSN: 2398-6352. DOI: 10.1038/s41746-018-0018-4. URL: https://doi.org/10.1038/s41746-018-0018-4.

[4] "Tukey's HSD Test". In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer New York, 2013, pp. 2303–2303. ISBN: 978-1-4419-9863-7. DOI: 10.1007/978-1-4419-9863-7_101573. URL: https://doi.org/10.1007/978-1-4419-9863-7_101573.

[5] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine." In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451. URL: https://doi.org/10.1214/aos/1013203451.

[6] Glassboc Medicine. *Measuring Performance: The Confusion Matrix*. [Online; accessed 15-August-2021]. URL: https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/.

[7] Isabelle Guyon et al. "Gene Selection for Cancer Classification using Support Vector Machines". In: *Machine Learning* 46.2 (2002), p. 1883. DOI: 10.1023/A:1012487302797. URL: https://doi.org/10.1023/A:1012487302797.

[8] Geoffrey E. Hinton. "Connectionist Learning Procedures". In: *Artif. Intell.* 40 (1989), pp. 185–234.

[9] Jason Brownlee. *Data Leakage in Machine Learning*. [Online; accessed 10-July-2021]. 2020. URL: https://machinelearningmastery.com/data-leakage-machine-learning/.

[10] Jason Brownlee. *How to Choose a Feature Selection Method For Machine Learning*. [Online; accessed 14-July-2021]. 2020. URL: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/.

[11] Jason Brownlee. *Overfitting and Underfitting With Machine Learning Algorithms*. [Online; accessed 10-July-2021]. 2019. URL: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

[12] H. Lam K. "Real-world keystroke dynamics are a potentially valid biomarker for clinical disability in multiple sclerosis." In: *Multiple Sclerosis Journal* (2020). DOI: 1352458520968797.

[13] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[14] Laurens van der Maaten. "Accelerating t-SNE using Tree-Based Algorithms". In: *Journal of Machine Learning Research* 15.93 (2014), pp. 3221–3245. URL: http://jmlr.org/papers/v15/vandermaaten14a.html.

[15] Matthew Stewart. *Understanding Dataset Shift*. [Online; accessed 04-September-2021]. URL: https://towardsdatascience.com/understanding-dataset-shift-f2a5a262a766.

[16] Neurocast B.V. *Neurocast*. [Online; accessed 02-June-2021]. 2016. URL: https://www.neurocast.nl/.

[17] Neurocast B.V. *Neurokeys*. [Online; accessed 02-June-2021]. 2016. URL: http://neurokeys.app//.

[18] OmniSci. *Feature Engineering Definition*. [Online; accessed 10-June-2021]. URL: https://www.omnisci.com/technical-glossary/feature-engineering.

[19] Karl Pearson. "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (July 1900), pp. 157–175. DOI: 10.1080/14786440009463897. URL: https://doi.org/10.1080/14786440009463897.

[20] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[21] L. E. Peterson. "K-nearest neighbor". In: *Scholarpedia* 4.2 (2009). revision #137311, p. 1883. DOI: 10.4249/scholarpedia.1883.

[22] Joshua S. Richman and J. Randall Moorman. "Physiological time-series analysis using approximate entropy and sample entropy". In: *American Journal of Physiology-Heart and Circulatory Physiology* 278.6 (2000). PMID: 10843903, H2039–H2049. DOI: 10.1152/ajpheart.2000.278.6.H2039. eprint: https://doi.org/10.1152/ajpheart.2000.278.6.H2039. URL: https://doi.org/10.1152/ajpheart.2000.278.6.H2039.

[23] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[24] J. Twose et al. "Early-warning signals for disease activity in patients diagnosed with multiple sclerosis based on keystroke dynamics". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.11 (2020), p. 113133. DOI: 10.1063/5.0022031. eprint: https://doi.org/10.1063/5.0022031. URL: https://doi.org/10.1063/5.0022031.

[25] Robert-Andrei Voicu et al. "Human Physical Activity Recognition Using Smartphone Sensors". In: *Sensors* 19.3 (2019). ISSN: 1424-8220. DOI: 10.3390/s19030458. URL: https://www.mdpi.com/1424-8220/19/3/458.

[26] He Yi and Li Ye. "Physical Activity Recognition Utilizing the Built-In Kinematic Sensors of a Smartphone". In: *International Journal of Distributed Sensor Networks* 9 (2013). ISSN: 2398-6352. DOI: 10.1038/s41746-019-0084-2. URL: https://doi.org/10.1038/s41746-019-0084-2.