D4.7: Behaviour change detection analysis prototype

Dissemination level:	Public
Document type:	Demonstrator
Version:	1.0.0
Date:	November 29, 2019

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Document Details

Project Number	769553
Project title	Council of Coaches
Title of deliverable	Behaviour change detection analysis prototype
Due date of deliverable	November 30, 2019
Work package	WP4
Author(s)	Oresti Banos (CMC), Kostas Konsolakis (CMC)
Reviewer(s)	Álvaro Fides (UPV), Harm op den Akker (RRD), Jorien van Loon (CMC)
Approved by	Coordinator
Dissemination level	Public
Document type	Demonstrator
Total number of pages	23

Partners

- University of Twente – Centre for Monitoring and Coaching (CMC)
- Roessingh Research and Development (RRD)
- Danish Board of Technology Foundation (DBT)
- Sorbonne University (SU)
- University of Dundee (UDun)
- Universitat Politècnica de València, Grupo SABIEN (UPV)
- Innovation Sprint (iSPRINT)

Abstract

This deliverable (D4.7) describes the software implementation of the methods developed in D4.6 for the detection of behaviour changes, presenting the technical setup and the scripts for detecting behaviour changes.

Table of Contents

List of figures

Symbols, abbreviations and acronyms

API	Application Programme Interface
CMC	Centre for Monitoring and Coaching
COUCH	Council of Coaches
CPD	Change Point Detection
D	Deliverable
EC	European Commission
GPS	Global Positioning System
HBAF	Holistic Behaviour Analysis Framework
HCI	Human-Computer Interaction
JSON	JavaScript Object Notification
M	Month
MS	Milestone
REST	Representational State Transfer
RRD	Roessingh Research and Development
SU	Sorbonne University
UPV	Universitat Politècnica de València
UT	University of Twente

1 Introduction

After presenting the methods used for detecting changes between time periods and determining the significance of the detected changes, that we presented in D4.6, this deliverable dives deeper into the software components needed for the characterisation and identification of behaviour changes. In particular, D4.7 accompanies D4.6 and aims to present the updated Holistic Behaviour Analysis Framework (HBAF) architecture (originally presented in D4.3), including the software components and the scripts for modelling behaviour changes.

First, the technical setup including the hardware components and the system's architecture is presented in Section 3. Then, the developed models for detecting and assessing behaviour changes are implemented in Section 4.

2 Objectives

The main objective of this deliverable (D4.7) is to describe the software implementation of the methods developed to detect behaviour changes based on short-term behaviour information (D4.6).

3 Technical Setup

3.1 Hardware

The hardware setup is based on the same approach as presented in D4.3 (please refer to Section 3.1 of that deliverable for a full description). Specifically, on-body and off-body sensor data such as acceleration, phone call logs and ambient sound are collected and stored into the Holistic Behaviour Analysis Framework (HBAF) in order to detect physical, social, emotional and cognitive short-term behaviours. Based on these behaviours, a number of features are extracted for identifying physical, social, emotional and cognitive behaviours changes over different periods of time.

3.2 Software

One of the key points of the HBAF is to analyse multimodal sensor data and extract useful information to describe the user's behaviour. In that spirit, once behaviour changes are detected they are constantly pushed through a secured connection to the Shared Knowledge Base, where they are stored and available for use in the Council of Coaches dialogues with the end-users. Thus, the provided information can be used in order to feed the dialogues among coaches and trigger possible updates to coaching strategies and actions. The complete operation flow of the HBAF is presented in the following picture (see Figure 1 below).

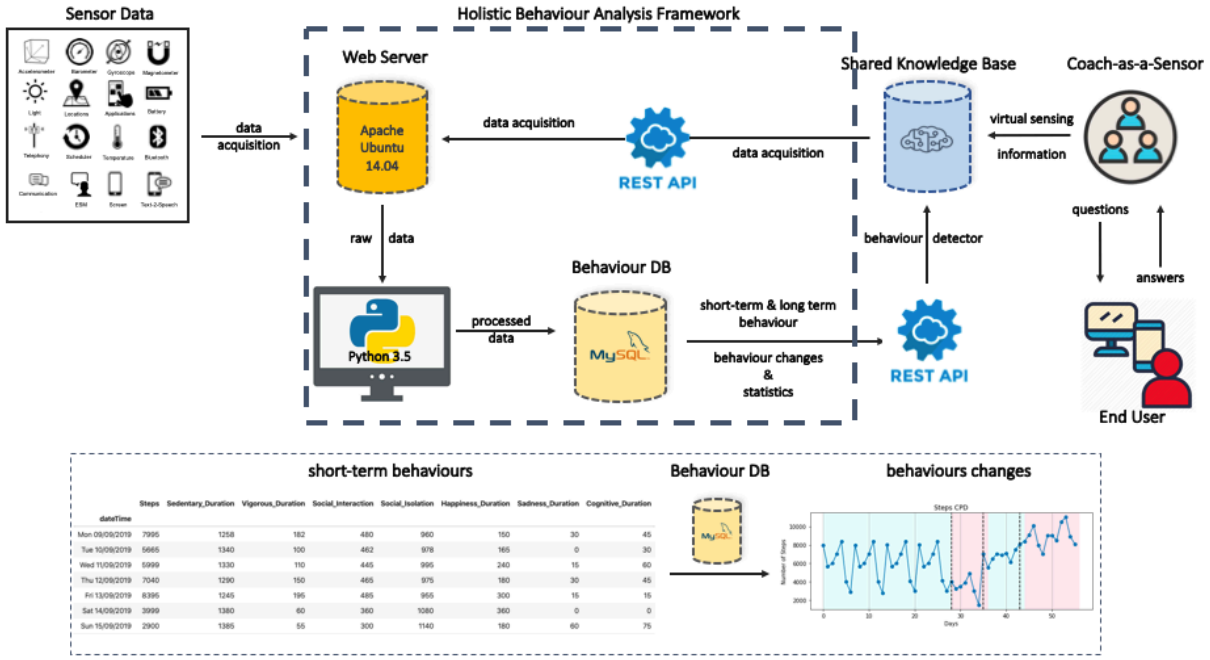

Figure 1: Operation flow of the HBAF (from raw sensor data to behaviour changes).

As it was first introduced in D4.3, the operation flow, which is extended here, unwraps as follows. Smartphone raw data such as accelerometer, GPS, Bluetooth and ambient noise are recorded and stored into the Apache Web server on a Linux system (Ubuntu, 2019). Then, scripts are developed in Python 3.5 (Python, 2015) in order to process the raw data and detect short-term and long-term behaviours but also behaviour changes. At first, short-term behaviour models detect physical, social, emotional and cognitive behaviours, which are stored into a MySQL 5.7 database (MySQL, 2019). Additionally, the generated short-term behaviours time series are further analysed in order to detect and assess potential behaviour changes. Specifically, as explained in D4.6, the Dynamic Programming Search Method (ENS Paris-Saclay, 2018) is used in order to estimate when, and for how long, a change occurs. Furthermore, some relevant statistics are extracted from these time series to determine the magnitude of the change (average values for each segment representing a change in the behaviour trend; day-to-day or week-to-week absolute difference - i.e. absolute change -; and their relative

difference - i.e. relative change -), which in turn may help identifying whether the detected change is worth reporting to the council of coaches to take further action (e.g. a specific intervention).

Then, the Shared Knowledge Base gets permanent access to this database via a secured REST API connection (Framework Slim, 2017), based on JSON web token authentication. The HBAF communicates with the Shared Knowledge Base through the cURL web2web communication protocol. The data are distributed through JSON format according to RFC 8259 (Datatracker, 2018) and ECMA-404 (ECMA, 2017). It is worth mentioning that the Shared Knowledge Base component can get access to the HBAF and post data extracted from the dialogs between user and coach (Coach-as-a-Sensor) for further analysis, as was presented in Section 3 of the D4.6.

The software scripts implemented for both the detection and assessment of behaviour changes are presented in Section 4. However, for the sake of clarity and contextualisation, we show some exemplary data frames resulting from the application of such scripts. First, we present in Figure 2 the input data, which is a subset of the dataset used for evaluation in D4.6 (see '4.1.1 Experimental Setup' in D4.6), including physical, social, emotional and cognitive short-term behaviours. In Figure 3, we present the absolute difference of two data points of the short-term behaviour dataset, comparing namely the values of every day of the week with the same day of the previous week (i.e. freq='7 days'). The absolute difference represents the deviation of the user's behaviour from one week to another, which can be considered as a quantifier of behaviour change. In Figure 4, we present the magnitude of the change in relative terms, which is attained by normalizing the absolute difference. The relative change can be used to quantify the change among all the behaviours, which will normally have a different value range. Finally, in Figure 5, we show for every behaviour domain and a given individual, the total number of changes, when these changes start and end, and also some relevant statistics that can be used during the Council of Coaches sessions in order to assess the estimated behaviour changes. For instance, user's physical behaviour time series analysis indicates 3 behaviour changes; the first starts on 7/10/2019 (user's retirement day), the second starts one week later on 15/10/2019 (when COUCH intervention takes place), and the third change starts on 23/10/2019. It is clear that the first physical behaviour change occurs because the user is significantly less active after the retirement day (compared to how he used to be during the working days), while the third physical behaviour change occurs because the user tries to maintain a healthier lifestyle (after the COUCH intervention).

short_term_behaviour - DataFrame

Index	Day	Week	Steps	Sedentary_Duration	Vigorous_Duration	Social_Interaction	Social_Isolation	Happiness_Duration	Sadness_Duration	Cognitive_Duration
2019-09-30 00:00:00	Monday	4	8002	1265	175	475	965	105	150	0
2019-10-01 00:00:00	Tuesday	4	5670	1330	110	435	1005	135	60	0
2019-10-02 00:00:00	Wednesday	4	6002	1320	120	465	975	120	110	45
2019-10-03 00:00:00	Thursday	4	7050	1290	150	485	955	105	85	0
2019-10-04 00:00:00	Friday	4	8400	1260	180	540	900	180	45	0
2019-10-05 00:00:00	Saturday	4	4150	1365	75	375	1065	265	60	45
2019-10-06 00:00:00	Sunday	4	3010	1395	45	345	1095	240	120	60
2019-10-07 00:00:00	Monday	5	4003	1380	60	270	1170	105	110	120
2019-10-08 00:00:00	Tuesday	5	3270	1395	45	245	1195	120	60	0
2019-10-09 00:00:00	Wednesday	5	3500	1385	55	295	1145	135	90	60
2019-10-10 00:00:00	Thursday	5	3900	1385	55	240	1200	105	85	0
2019-10-11 00:00:00	Friday	5	4900	1340	100	295	1145	165	75	0
2019-10-12 00:00:00	Saturday	5	3000	1390	50	355	1085	240	60	0
2019-10-13 00:00:00	Sunday	5	1500	1420	20	315	1125	210	120	75
2019-10-14 00:00:00	Monday	6	7003	1290	150	310	1130	150	75	15
2019-10-15 00:00:00	Tuesday	6	5550	1330	110	295	1145	180	45	0
2019-10-16 00:00:00	Wednesday	6	6500	1300	140	305	1135	165	75	60
2019-10-17 00:00:00	Thursday	6	7055	1285	155	300	1140	180	60	45
2019-10-18 00:00:00	Friday	6	6900	1295	145	325	1115	210	60	30
2019-10-19 00:00:00	Saturday	6	7100	1285	155	355	1085	240	45	60
2019-10-20 00:00:00	Sunday	6	6150	1310	130	345	1095	225	90	90

Figure 2: Short-term Behaviour dataset.

behaviour_change_absolute - DataFrame

Index	Day	Week	Steps	Sedentary_Duration	Vigorous_Duration	Social_Interaction	Social_Isolation	Happiness_Duration	Sadness_Duration	Cognitive_Duration
2019-09-30 00:00:00	Monday	4	1	5	5	10	10	30	105	60
2019-10-01 00:00:00	Tuesday	4	1	0	0	15	15	45	60	0
2019-10-02 00:00:00	Wednesday	4	1	5	5	70	70	0	80	45
2019-10-03 00:00:00	Thursday	4	5	0	0	30	30	0	70	0
2019-10-04 00:00:00	Friday	4	10	10	10	45	45	60	15	60
2019-10-05 00:00:00	Saturday	4	50	5	5	20	20	35	45	15
2019-10-06 00:00:00	Sunday	4	10	10	10	30	30	30	60	0
2019-10-07 00:00:00	Monday	5	3999	115	115	205	205	0	40	120
2019-10-08 00:00:00	Tuesday	5	2400	65	65	190	190	15	0	0
2019-10-09 00:00:00	Wednesday	5	2502	65	65	170	170	15	20	15
2019-10-10 00:00:00	Thursday	5	3150	95	95	245	245	0	0	0
2019-10-11 00:00:00	Friday	5	3500	80	80	245	245	15	30	0
2019-10-12 00:00:00	Saturday	5	1150	25	25	20	20	25	0	45
2019-10-13 00:00:00	Sunday	5	1510	25	25	30	30	30	0	15
2019-10-14 00:00:00	Monday	6	3000	90	90	40	40	45	35	105
2019-10-15 00:00:00	Tuesday	6	2280	65	65	50	50	60	15	0
2019-10-16 00:00:00	Wednesday	6	3000	85	85	10	10	30	15	0
2019-10-17 00:00:00	Thursday	6	3155	100	100	60	60	75	25	45
2019-10-18 00:00:00	Friday	6	2000	45	45	30	30	45	15	30
2019-10-19 00:00:00	Saturday	6	4100	105	105	0	0	0	15	60
2019-10-20 00:00:00	Sunday	6	4650	110	110	30	30	15	30	15

Figure 3: Dataset for behaviour change based on the absolute difference of two elements (freq='7 days').

behaviour_change_relative - DataFrame

Index	Day	Week	Steps	Sedentary_Duration	Vigorous_Duration	Social_Interaction	Social_Isolation	Happiness_Duration	Sadness_Duration	Cognitive_Duration
2019-09-30 00:00:00	Monday	4	0	0.0434783	0.0434783	0.0408163	0.0408163	0.166667	1	0.5
2019-10-01 00:00:00	Tuesday	4	0	0	0	0.0612245	0.0612245	0.25	0.571429	0
2019-10-02 00:00:00	Wednesday	4	0	0.0434783	0.0434783	0.285714	0.285714	0	0.761905	0.375
2019-10-03 00:00:00	Thursday	4	0.0008604	0	0	0.122449	0.122449	0	0.666667	0
2019-10-04 00:00:00	Friday	4	0.0019359	0.0869565	0.0869565	0.183673	0.183673	0.333333	0.142857	0.5
2019-10-05 00:00:00	Saturday	4	0.0105399	0.0434783	0.0434783	0.0816327	0.0816327	0.194444	0.428571	0.125
2019-10-06 00:00:00	Sunday	4	0.0019359	0.0869565	0.0869565	0.122449	0.122449	0.166667	0.571429	0
2019-10-07 00:00:00	Monday	5	0.85997	1	1	0.836735	0.836735	0	0.380952	1
2019-10-08 00:00:00	Tuesday	5	0.516025	0.565217	0.565217	0.77551	0.77551	0.0833333	0	0
2019-10-09 00:00:00	Wednesday	5	0.537965	0.565217	0.565217	0.693878	0.693878	0.0833333	0.190476	0.125
2019-10-10 00:00:00	Thursday	5	0.67735	0.826087	0.826087	1	1	0	0	0
2019-10-11 00:00:00	Friday	5	0.752635	0.695652	0.695652	1	1	0.0833333	0.285714	0
2019-10-12 00:00:00	Saturday	5	0.24715	0.217391	0.217391	0.0816327	0.0816327	0.138889	0	0.375
2019-10-13 00:00:00	Sunday	5	0.324586	0.217391	0.217391	0.122449	0.122449	0.166667	0	0.125
2019-10-14 00:00:00	Monday	6	0.645085	0.782609	0.782609	0.163265	0.163265	0.25	0.333333	0.875
2019-10-15 00:00:00	Tuesday	6	0.490213	0.565217	0.565217	0.204082	0.204082	0.333333	0.142857	0
2019-10-16 00:00:00	Wednesday	6	0.645085	0.73913	0.73913	0.0408163	0.0408163	0.166667	0.142857	0
2019-10-17 00:00:00	Thursday	6	0.678425	0.869565	0.869565	0.244898	0.244898	0.416667	0.238095	0.375
2019-10-18 00:00:00	Friday	6	0.429985	0.391304	0.391304	0.122449	0.122449	0.25	0.142857	0.25
2019-10-19 00:00:00	Saturday	6	0.881695	0.913043	0.913043	0	0	0	0.142857	0.5
2019-10-20 00:00:00	Sunday	6	1	0.956522	0.956522	0.122449	0.122449	0.0833333	0.285714	0.125

Figure 4: Dataset for behaviour change based on the normalized absolute differences.

behaviour_change_quantifier - DataFrame

Index	User_ID	Behaviour_Type	Change	Timestamp_Start	Timestamp_End	Total_Duration(Days)	Mean	Diff	Relative_Diff
0	user01	Steps	0	2019-09-09 00:00:00	2019-10-06 00:00:00	28	6015.54	nan	nan
1	user01	Steps	1	2019-10-07 00:00:00	2019-10-14 00:00:00	8	3884.5	-2131.04	35.4255
2	user01	Steps	2	2019-10-15 00:00:00	2019-10-22 00:00:00	8	6856.88	2972.38	-76.5189
3	user01	Steps	3	2019-10-23 00:00:00	2019-11-03 00:00:00	12	8971	2114.12	-30.8322
4	user01	Sedentary_Duration	0	2019-09-09 00:00:00	2019-10-06 00:00:00	28	1316.36	nan	nan
5	user01	Sedentary_Duration	1	2019-10-07 00:00:00	2019-10-14 00:00:00	8	1373.12	56.7679	-4.3125
6	user01	Sedentary_Duration	2	2019-10-15 00:00:00	2019-10-22 00:00:00	8	1291.88	-81.25	5.91716
7	user01	Sedentary_Duration	3	2019-10-23 00:00:00	2019-11-03 00:00:00	12	1229.17	-62.7083	4.85406
8	user01	Vigorous_Duration	0	2019-09-09 00:00:00	2019-10-06 00:00:00	28	123.643	nan	nan
9	user01	Vigorous_Duration	1	2019-10-07 00:00:00	2019-10-14 00:00:00	8	66.875	-56.7679	45.9128
10	user01	Vigorous_Duration	2	2019-10-15 00:00:00	2019-10-22 00:00:00	8	148.125	81.25	-121.495
11	user01	Vigorous_Duration	3	2019-10-23 00:00:00	2019-11-03 00:00:00	12	210.833	62.7083	-42.3347
12	user01	Social_Interaction	0	2019-09-09 00:00:00	2019-10-06 00:00:00	28	433.179	nan	nan
13	user01	Social_Interaction	1	2019-10-07 00:00:00	2019-10-10 00:00:00	4	262.5	-170.679	39.4014
14	user01	Social_Interaction	2	2019-10-11 00:00:00	2019-10-18 00:00:00	8	312.5	50	-19.0476
15	user01	Social_Interaction	3	2019-10-19 00:00:00	2019-11-03 00:00:00	16	348.688	36.1875	-11.58
16	user01	Social_Isolation	0	2019-09-09 00:00:00	2019-10-06 00:00:00	28	1006.82	nan	nan
17	user01	Social_Isolation	1	2019-10-07 00:00:00	2019-10-10 00:00:00	4	1177.5	170.679	-16.9522
18	user01	Social_Isolation	2	2019-10-11 00:00:00	2019-10-18 00:00:00	8	1127.5	-50	4.24628
19	user01	Social_Isolation	3	2019-10-19 00:00:00	2019-11-03 00:00:00	16	1091.31	-36.1875	3.20953
20	user01	Happiness_Duration	0	2019-09-09 00:00:00	2019-10-06 00:00:00	28	202.857	nan	nan
21	user01	Happiness_Duration	1	2019-10-07 00:00:00	2019-10-10 00:00:00	4	116.25	-86.6071	42.6937
22	user01	Happiness_Duration	2	2019-10-11 00:00:00	2019-10-22 00:00:00	12	202.5	86.25	-74.1935
23	user01	Happiness_Duration	3	2019-10-23 00:00:00	2019-11-03 00:00:00	12	287.833	85.3333	-42.1399
24	user01	Sadness_Duration	0	2019-09-09 00:00:00	2019-09-28 00:00:00	20	21	nan	nan
25	user01	Sadness_Duration	1	2019-09-29 00:00:00	2019-10-14 00:00:00	16	85.3125	64.3125	-306.25
26	user01	Sadness_Duration	2	2019-10-15 00:00:00	2019-10-22 00:00:00	8	58.125	-27.1875	31.8681
27	user01	Sadness_Duration	3	2019-10-23 00:00:00	2019-11-03 00:00:00	12	32.5833	-25.5417	43.9427
28	user01	Cognitive_Duration	0	2019-09-09 00:00:00	2019-09-16 00:00:00	8	41.25	nan	nan
29	user01	Cognitive_Duration	1	2019-09-17 00:00:00	2019-10-14 00:00:00	28	27.3214	-13.9286	33.7662
30	user01	Cognitive_Duration	2	2019-10-15 00:00:00	2019-10-22 00:00:00	8	46.875	19.5536	-71.5686
31	user01	Cognitive_Duration	3	2019-10-23 00:00:00	2019-11-03 00:00:00	12	85	38.125	-81.3333

Figure 5: Dataset for quantifying and assessing behaviour changes.

4 Scripts for Behaviour Change Detection

The following script describes the methodology for detecting and assessing behaviour changes. First, the short-term behaviour data is fetched from the HBAF database for a given time span (this period can be configured and may change from subject to subject). Then, the behaviour changes are detected through the dynamic programming search method. This method is implemented using the Ruptures Python library (ENS Paris-Saclay, 2018). Finally, the segments that represent a behaviour change, and the absolute and relative change are computed for every segment. Further details are provided along with the code in the accompanying comments (please see below).

```
# ================================================================================
# Behaviour Change Model
# ================================================================================

#import libraries
import itertools
import pandas as pd
import numpy as np
from pandas import datetime
import os
import os.path
import statsmodels.api as sm
import matplotlib.pyplot as plt
import ruptures as rpt
from pandas.plotting import register_matplotlib_converters
from ruptures.metrics import precision_recall
from ruptures.metrics import randindex
from ruptures.metrics import hausdorff
from os import listdir
import mysql.connector
from scipy.signal import butter, lfilter,argrelmax
import math
from datetime import datetime, timedelta


# ================================================================================
#    Timestamp Condition for data acquisition
# ================================================================================
#defines the timestamp query in order to get the available data from database
date_threshold1 = '2019-09-09'

# ================================================================================
# list with available userID
# ================================================================================
list_userID = ['user01', 'user02', 'user03']


#================================================================================
#     START CONNECTION and Read Short_Term_Behaviour database
#================================================================================
for userID in list_userID:

    # Open database connection
    #define server name and password!
    db = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',passwd='pswd',db='
ShortTerm_Behaviour' )


    # prepare a cursor object using cursor() method
    cursor = db.cursor()

```

```
52.     sql_df_short = "SELECT * FROM ShortTerm_Behaviour.Short_Term_Behaviour WHERE timestamp>=%s
    "%date_threshold1 + " AND user_id='%s'"%userID
53.     short_term_behaviour = pd.read_sql(sql_df_short, db)
54.
55.     db.close()
56.
57. # ================================================================================
58. # SQL Connection - Create a table for the first time (if there is not)
59. # ================================================================================
60.     create_table() #method for creating a table for the first time (if there is not)
61.
62.
63. # ================================================================================
64. # SQL Connection - Insert Data
65. # ================================================================================
66.
67.     #returns a dataset with the segments of the detected behaviour changes
68.     behaviour_change = Behaviour_Change(short_term_behaviour, userID)
69.
70.     #returns two datasets, with the absolute difference and relative absolute difference of be
    haviour change
71.     behaviour_change_absolute, behaviour_change_relative  = Absolute_Relative_Change(short_ter
    m_behaviour, userID)
72.
73.     #returns a dataset with the weekly percentage change of user's behaviour
74.     behaviour_change_pct = PCT_Change(short_term_behaviour, userID)
75.
76.     #returns a dataset in order to quantify and assess the behaviour changes
77.     behaviour_change_quantifier = Quantifier_Behaviour_Change(short_term_behaviour, behaviour_
    change, userID)
78.
79.     connect = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',passwd='pswd'
    ,db='ShortTerm_Behaviour' )
80.
81.     cursor = connect.cursor()
82.
83.     #convert NaN values to None numeric values for SQL
84.     behaviour_change = behaviour_change.where(pd.notnull(behaviour_change), None)
85.     behaviour_change_absolute = behaviour_change_absolute.where(pd.notnull(behaviour_change_ab
    solute), None)
86.     behaviour_change_relative = behaviour_change_relative.where(pd.notnull(behaviour_change_re
    lative), None)
87.     behaviour_change_pct = behaviour_change_pct.where(pd.notnull(behaviour_change_pct), None)

88.     behaviour_change_quantifier = behaviour_change_quantifier.where(pd.notnull(behaviour_chang
    e_quantifier), None)
89.
90.     #inserts data rows to the Behaviour_Change table on HBAF database
91.     for index,row in behaviour_change.iterrows():
92.
93.         sql="""INSERT INTO Behaviour_Change(Timestamp_Start, Key_id, User_id, Steps, Sedentary
    _Duration, Vigorous_Duration, Social_Interaction,
94.         Social_Isolation, Happiness_Duration, Sadness_Duration, Cognitive_Duration)
95.         SELECT  %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s FROM DUAL
96.         WHERE NOT EXISTS (SELECT Key_id FROM Behaviour_Change WHERE Key_id=%s); """
97.
98.         values = [row['Timestamp_Start'], row['Key_id'], row['User_ID'], row['Steps'], row['Se
    dentary_Duration'], row['Vigorous_Duration'], row['Social_Interaction'],
99.                   row['Social_Isolation'], row['Happiness_Duration'], row['Sadness_Duration'],
     row['Cognitive_Duration'], row['Key_id'] ]
100.        cursor.execute(sql,values)
101.        connect.commit()
102.
103.    #inserts data rows to the Behaviour_Change_Absolute table on HBAF database
104.    for index,row in behaviour_change_absolute.iterrows():
105.
```

```
106.        sql="""INSERT INTO Behaviour_Change_Absolute(Timestamp_Start, Key_id, User_id, Steps,
    Sedentary_Duration, Vigorous_Duration, Social_Interaction,
107.        Social_Isolation, Happiness_Duration, Sadness_Duration, Cognitive_Duration)
108.        SELECT  %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s FROM DUAL
109.        WHERE NOT EXISTS (SELECT Key_id FROM Behaviour_Change_Absolute WHERE Key_id=%s); """
110.
111.        values = [row['Timestamp_Start'], row['Key_id'], row['User_ID'], row['Steps'], row['Se
    dentary_Duration'], row['Vigorous_Duration'], row['Social_Interaction'],
112.                  row['Social_Isolation'], row['Happiness_Duration'], row['Sadness_Duration'],
     row['Cognitive_Duration'], row['Key_id'] ]
113.        cursor.execute(sql,values)
114.        connect.commit()
115.
116.    #inserts data rows to the Behaviour_Change_Relative table on HBAF database
117.    for index,row in behaviour_change_relative.iterrows():
118.
119.        sql="""INSERT INTO Behaviour_Change_Relative(Timestamp_Start, Key_id, User_id, Steps,
    Sedentary_Duration, Vigorous_Duration, Social_Interaction,
120.        Social_Isolation, Happiness_Duration, Sadness_Duration, Cognitive_Duration)
121.        SELECT  %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s FROM DUAL
122.        WHERE NOT EXISTS (SELECT Key_id FROM Behaviour_Change_Relative WHERE Key_id=%s); """
123.
124.        values = [row['Timestamp_Start'], row['Key_id'], row['User_ID'], row['Steps'], row['Se
    dentary_Duration'], row['Vigorous_Duration'], row['Social_Interaction'],
125.                  row['Social_Isolation'], row['Happiness_Duration'], row['Sadness_Duration'],
     row['Cognitive_Duration'], row['Key_id'] ]
126.        cursor.execute(sql,values)
127.        connect.commit()
128.
129.    #inserts data rows to the Behaviour_Change_Relative table on HBAF database
130.    for index,row in behaviour_change_pct.iterrows():
131.
132.        sql="""INSERT INTO Behaviour_Change_PCT(Timestamp_Start, Key_id, User_id, Steps, Seden
    tary_Duration, Vigorous_Duration, Social_Interaction,
133.        Social_Isolation, Happiness_Duration, Sadness_Duration, Cognitive_Duration)
134.        SELECT  %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s FROM DUAL
135.        WHERE NOT EXISTS (SELECT Key_id FROM Behaviour_Change_Relative WHERE Key_id=%s); """
136.
137.        values = [row['Timestamp_Start'], row['Key_id'], row['User_ID'], row['Steps'], row['Se
    dentary_Duration'], row['Vigorous_Duration'], row['Social_Interaction'],
138.                  row['Social_Isolation'], row['Happiness_Duration'], row['Sadness_Duration'],
     row['Cognitive_Duration'], row['Key_id'] ]
139.        cursor.execute(sql,values)
140.        connect.commit()
141.
142.    #inserts data rows to the Behaviour_Change_Quantifier table on HBAF database
143.    for index,row in behaviour_change_quantifier.iterrows():
144.        sql="""INSERT INTO Behaviour_Change_Quantifier(Key_id, User_id, Behaviour_Type, Change
    _Num, Timestamp_Start, Timestamp_End, Total_Duration,
145.        Mean, Diff, Relative_Diff)
146.        SELECT  %s, %s, %s, %s, %s, %s, %s, %s, %s, %s FROM DUAL
147.        WHERE NOT EXISTS (SELECT Key_id FROM Behaviour_Change_Quantifier WHERE Key_id=%s); """

148.
149.        values = [row['Key_id'], row['User_ID'], row['Behaviour_Type'], row['Change'], row['Ti
    mestamp_Start'], row['Timestamp_End'],
150.                  row['Total_Duration(Days)'], row['Mean'], row['Diff'], row['Relative_Diff'],
     row['Key_id'] ]
151.        cursor.execute(sql,values)
152.        connect.commit()
153.
154.
155.    print (behaviour_change['Key_id'][-
    1], 'and %s rows inserted / Behaviour_Change DB is ok'%len(behaviour_change))
```

```
    print (behaviour_change_absolute['Key_id'][-1], 'and %s rows inserted / Behaviour_Change_Absolute DB is ok'%len(behaviour_change_absolute))
    print (behaviour_change_relative['Key_id'][-1], 'and %s rows inserted / Behaviour_Change_Relative DB is ok'%len(behaviour_change_relative))
    print (behaviour_change_pct['Key_id'][-1], 'and %s rows inserted / Behaviour_Change_PCT DB is ok'%len(behaviour_change_pct))
    print (behaviour_change_quantifier['Key_id'][-1], 'and %s rows inserted / Behaviour_Change_Quantifier DB is ok'%len(behaviour_change_quantifier))

    cursor.close()
    connect.close()
#userID for loop ends - all data are calculated/inserted for every user



# ================================================================================
# Functions for Change Point Detection
# ================================================================================

#returns the number of changes based on the Dynamic Programming Search Method
def CPD_Dynamic_Programming(df_short, plot_option=True):
    df_short = df_short.dropna() #removes NaN values
    points=np.array(df_short)
    model = "l1"
    algo = rpt.Dynp(model=model, min_size=2, jump=4).fit(points)
    my_bkps = algo.predict(n_bkps=3)
    if plot_option == True:
        CPD_Plot (points, my_bkps)
    return my_bkps

#plots the CPD figures
def CPD_Plot (points, my_bkps):
    rpt.display(points, my_bkps, figsize=(13, 5))
    plt.title('CPD: Dynamic Programming Search Method')
    plt.xlabel('Days')
    plt.ylabel('Percentage Change')
    plt.gca().xaxis.grid(True)
    plt.show()


#returns a dataframe with the number of behaviour changes
def Behaviour_Change (df_short, userID):
    # The short_term_behaviour dataframe was created for D4.3 and contains the short-term physical, social, emotional and cognitive behaviours per day
    df_change = df_short.copy()
    # The behaviour_change is based on the columns of the short_term_behaviour dataframe, which are initially edited to nan values.
    df_change[df_change.columns[2:]] = np.nan

    for i in range(2,len(df_change.columns)):  #the range=2 selects the columns from Steps and onwards
        print (df_change.columns[i])
        #calculates the number of changes and their indexes
        bkps = CPD_Dynamic_Programming(df_short.iloc[:,i])
        for change in range(1, len(bkps)):
            print (change)
            if change-1 < (len(bkps)):
                #it adds the number of change when this occurs
                df_change.iloc[ bkps[change-1] : bkps[change] , i] = change

```

```
    df_change["Timestamp_Start"] = df_change.index
    df_change["User_ID"] = userID
    df_change["period"] = df_change.index.strftime('%s')
    df_change["Key_id"] = df_change[['period', 'User_ID']].apply(lambda x: '_'.join(x), axis=1
)
    return df_change

#returns two dataframes with the number of absolute and relative behaviour changes
def Absolute_Relative_Change(df_short, userID):
    df_Abs_Change, df_Relative_Change = df_short.copy(), df_short.copy()
    df_Abs_Change[df_Abs_Change.columns[2:]] = np.nan
    df_Relative_Change[df_Relative_Change.columns[2:]] = np.nan

    #calculates the diff data based on freq='7D'
    df_Abs_Change.iloc[:,2:] = df_short.iloc[:,2:].diff(periods=7).abs()

    #calculates the normalized df_Abs_Change
    df_Relative_Change.iloc[:,2:] = (df_Abs_Change.iloc[:,2:] - df_Abs_Change.iloc[:,2:].min()
)/(df_Abs_Change.iloc[:,2:].max() - df_Abs_Change.iloc[:,2:].min())
    df_Relative_Change["Timestamp_Start"] = df_Relative_Change.index
    df_Relative_Change["User_ID"] = userID
    df_Relative_Change["period"] = df_Relative_Change.index.strftime('%s')
    df_Relative_Change["Key_id"] = df_Relative_Change[['period', 'User_ID']].apply(lambda x: '
_'.join(x), axis=1)

    #extra columns for df_Abs_Change
    df_Abs_Change["Timestamp_Start"] = df_Abs_Change.index
    df_Abs_Change["User_ID"] = userID
    df_Abs_Change["period"] = df_Abs_Change.index.strftime('%s')
    df_Abs_Change["Key_id"] = df_Abs_Change[['period', 'User_ID']].apply(lambda x: '_'.join(x)
, axis=1)

    return df_Abs_Change, df_Relative_Change

#returns a dataframe based on the weekly percentage of change (freq= '7D')
def PCT_Change(df_short, userID):
    #selects the data based on freq='7D'

    df_PCT = df_short.copy()
    df_PCT[df_PCT.columns[2:]] = np.nan

    #loop starting from 2 (selects the columns from Steps and onwards)
    for i in range(2,len(df_PCT.columns)):
        #calculates the weekly PCT change
        df_PCT.iloc[:,i] = df_short.iloc[:,i].pct_change(freq='7D')

        #the following steps use interpolation approach in order to deal with the zero divisio
n problem (values such as cognitive duration might be 0)
        #replaces the nan values with 0, exluding the first 7 rows with nan values(due to week
ly PCT calculation)
        df_PCT.iloc[7:,i] = df_PCT.iloc[7:,i].fillna(0)
        #replaces the inf values (this results from division with 0) with nan values
        df_PCT.iloc[7:,i] = df_PCT.iloc[7:,i].replace([np.inf, -np.inf], np.nan)
        #interpolates nan values
        df_PCT.iloc[7:,i] = df_PCT.iloc[7:,i].interpolate(method='polynomial', order=2)

        #nornalization per column
        df_PCT.iloc[:,i]=(df_PCT.iloc[:,i]-df_PCT.iloc[:,i].min())/(df_PCT.iloc[:,i].max()-
df_PCT.iloc[:,i].min())

        #add extra columns
        df_PCT["Timestamp_Start"] = df_PCT.index
        df_PCT["User_ID"] = userID
        df_PCT["period"] = df_PCT.index.strftime('%s')
        df_PCT["Key_id"] = df_PCT[['period', 'User_ID']].apply(lambda x: '_'.join(x), axis=1)
```

```
270.
271.     return df_PCT
272.
273.
274.
275.
276.#returns the total number of changes, when these changes start and end, but also some relevant
     statistics (mean, diff, and relative_diff)
277.#that can be used during the COUCH sessions in order to assess the estimated behaviour changes
    .
278.def Quantifier_Behaviour_Change(df_short, df_change, userID):
279.
280.     df_change = df_change.fillna(0)
281.     cols = ['User_ID', 'Behaviour_Type', 'Change', 'Timestamp_Start', 'Timestamp_End', 'Total_
    Duration(Days)', 'Mean', 'Diff', 'Relative_Diff' ]
282.     #creates the behaviour_change_quantifier dataframe
283.     quantifier_behaviour_change = pd.DataFrame(columns = cols)
284.
285.     for col in range(2, len(df_change.columns)-4): #condition for only the necessary columns
286.         for i in range(1, len(df_change)):
287.             if df_change.fillna(0).iloc[i,col] == 0 and df_change.iloc[i,col] != df_change.ilo
    c[i+1,col]:
288.                 print ('No change')
289.                 #current condition df segment
290.                 mask1a = df_short.iloc[:,col].loc[df_change.iloc[:,col] == int(df_change.iloc[
    i,col])]
291.                 #next df segment
292.                 mask1b = df_short.iloc[:,col].loc[df_change.iloc[:,col] == int(df_change.iloc[
    i+1,col])]
293.                 diff1 = mask1b.mean() - mask1a.mean()
294.                 rel_diff1 = (mask1a.mean() - mask1b.mean()) / mask1a.mean() *100
295.                 print ('duration: start on ', mask1a.index[0], ' end on ', mask1a.index[-1])
296.                 print ('mean number: ', mask1a.mean())
297.                 quantifier_behaviour_change = quantifier_behaviour_change.append({'User_ID':'u
    ser01', 'Behaviour_Type': df_change.columns[col], 'Change':0,
298.                                                                                 'Timestamp_S
    tart':mask1a.index[0], 'Timestamp_End':mask1a.index[-
    1], 'Total_Duration(Days)':len(mask1a), 'Mean':mask1a.mean(),
299.                                                                                 'Diff':np.na
    n, 'Relative_Diff':np.nan}, ignore_index=True)
300.
301.             if df_change.fillna(0).iloc[i,col] > 0 and df_change.iloc[i-
    1,col] != df_change.iloc[i,col]:
302.
303.                 print ('change: ', int(df_change.iloc[i,col]))
304.                 #previous df segment
305.                 mask2a = df_short.iloc[:,col].loc[df_change.iloc[:,col] == int(df_change.iloc[
    i-1,col])]
306.                 #current condition df segment
307.                 mask2b = df_short.iloc[:,col].loc[df_change.iloc[:,col] == int(df_change.iloc[
    i,col])]
308.                 diff2 = mask2b.mean() - mask2a.mean()
309.                 rel_diff2 = (mask2a.mean() - mask2b.mean()) / mask2a.mean() *100
310.                 print ('duration: start on ', mask2b.index[0], ' end on ', mask2b.index[-1])
311.                 print ('mean number: ', mask2b.mean())
312.                 print ('diff number: ', diff2)
313.                 print ('relative diff number: ', rel_diff2)
314.
315.                 quantifier_behaviour_change = quantifier_behaviour_change.append({'User_ID':'u
    ser01','Behaviour_Type': df_change.columns[col], 'Change':int(behaviour_change.iloc[i,col]),
316.                                                                                 'Timestamp_S
    tart':mask2b.index[0], 'Timestamp_End':mask2b.index[-
    1],'Total_Duration(Days)':len(mask2b), 'Mean':mask2b.mean(),
317.                                                                                 'Diff':diff2
    , 'Relative_Diff':rel_diff2}, ignore_index=True)
318.
```

```
319.
320.    quantifier_behaviour_change['Change_id'] = "Change"+ quantifier_behaviour_change['Change']
    .apply(str)
321.    quantifier_behaviour_change['Key_id'] = quantifier_behaviour_change[['User_ID', 'Behaviour
    _Type', 'Change_id']].apply(lambda x: '_'.join(x), axis=1)
322.
323.    return quantifier_behaviour_change
324.
325.
326.
327.
328.
329.
330.
331.# ================================================================================
332.# Function for SQL Connection - Table Create if not exists
333.# ================================================================================
334.def create_table():
335.    #Creates the Behaviour_Change Table
336.    connectionObject = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',pass
    wd='pswd',db='ShortTerm_Behaviour' )
337.    try:
338.        # Create a cursor object
339.        cursorObject = connectionObject.cursor()
340.        sqlQuery = """CREATE TABLE IF NOT EXISTS Behaviour_Change (ID int(11) NOT NULL AUTO_IN
    CREMENT, Timestamp_Start datetime NOT NULL, Key_id varchar(150) NOT NULL,
341.        User_id varchar(150) NOT NULL, Steps int(11) DEFAULT NULL, Sedentary_Duration int(11)
    DEFAULT NULL, Vigorous_Duration int(11) DEFAULT NULL,
342.        Social_Interaction int(11) DEFAULT NULL, Social_Isolation int(11) DEFAULT NULL, Happin
    ess_Duration int(11) DEFAULT NULL,
343.        Sadness_Duration int(11) DEFAULT NULL, Cognitive_Duration int(11) DEFAULT NULL, UNIQUE
     KEY ID (ID), PRIMARY KEY (Key_id)) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1"""

344.
345.        # Execute the sqlQuery
346.        cursorObject.execute(sqlQuery)
347.        # SQL query string
348.        sqlQuery = "show tables"
349.        # Execute the sqlQuery
350.        cursorObject.execute(sqlQuery)
351.        #Fetch all the rows
352.        rows = cursorObject.fetchall()
353.
354.        for row in rows:
355.            print(row)
356.
357.    except Exception as e:
358.
359.        print("Exeception occured:{}".format(e))
360.
361.    finally:
362.
363.        connectionObject.close()
364.        cursorObject.close()
365.
366.    #Creates the Behaviour_Change_Absolute Table
367.    connectionObject = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',pass
    wd='pswd',db='ShortTerm_Behaviour' )
368.    try:
369.        # Create a cursor object
370.        cursorObject = connectionObject.cursor()
371.        sqlQuery = """CREATE TABLE IF NOT EXISTS Behaviour_Change_Absolute (ID int(11) NOT NUL
    L AUTO_INCREMENT, Timestamp_Start datetime NOT NULL, Key_id varchar(150) NOT NULL,
372.        User_id varchar(150) NOT NULL, Steps float(18) DEFAULT NULL, Sedentary_Duration float(
    18) DEFAULT NULL, Vigorous_Duration float(18) DEFAULT NULL,
```

```
373.        Social_Interaction float(18) DEFAULT NULL, Social_Isolation float(18) DEFAULT NULL, Ha
    ppiness_Duration float(18) DEFAULT NULL,
374.        Sadness_Duration float(18) DEFAULT NULL, Cognitive_Duration float(18) DEFAULT NULL, UN
    IQUE KEY ID (ID), PRIMARY KEY (Key_id)) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1"
    ""
375.
376.        # Execute the sqlQuery
377.        cursorObject.execute(sqlQuery)
378.        # SQL query string
379.        sqlQuery = "show tables"
380.        # Execute the sqlQuery
381.        cursorObject.execute(sqlQuery)
382.        #Fetch all the rows
383.        rows = cursorObject.fetchall()
384.
385.        for row in rows:
386.            print(row)
387.
388.    except Exception as e:
389.
390.        print("Exeception occured:{}".format(e))
391.
392.    finally:
393.
394.        connectionObject.close()
395.        cursorObject.close()
396.
397.
398.    #Creates the Behaviour_Change_Relative Table
399.    connectionObject = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',pass
    wd='pswd',db='ShortTerm_Behaviour' )
400.    try:
401.        # Create a cursor object
402.        cursorObject = connectionObject.cursor()
403.        sqlQuery = """CREATE TABLE IF NOT EXISTS Behaviour_Change_Relative (ID int(11) NOT NUL
    L AUTO_INCREMENT, Timestamp_Start datetime NOT NULL, Key_id varchar(150) NOT NULL,
404.        User_id varchar(150) NOT NULL, Steps float(18) DEFAULT NULL, Sedentary_Duration float(
    18) DEFAULT NULL, Vigorous_Duration float(18) DEFAULT NULL,
405.        Social_Interaction float(18) DEFAULT NULL, Social_Isolation float(18) DEFAULT NULL, Ha
    ppiness_Duration float(18) DEFAULT NULL,
406.        Sadness_Duration float(18) DEFAULT NULL, Cognitive_Duration float(18) DEFAULT NULL, UN
    IQUE KEY ID (ID), PRIMARY KEY (Key_id)) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1"
    ""
407.
408.        # Execute the sqlQuery
409.        cursorObject.execute(sqlQuery)
410.        # SQL query string
411.        sqlQuery = "show tables"
412.        # Execute the sqlQuery
413.        cursorObject.execute(sqlQuery)
414.        #Fetch all the rows
415.        rows = cursorObject.fetchall()
416.
417.        for row in rows:
418.            print(row)
419.
420.    except Exception as e:
421.
422.        print("Exeception occured:{}".format(e))
423.
424.    finally:
425.
426.        connectionObject.close()
427.        cursorObject.close()
428.
429.    #Creates the Behaviour_Change_PCT Table
```

```
430.    connectionObject = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',pass
    wd='pswd',db='ShortTerm_Behaviour' )
431.    try:
432.        # Create a cursor object
433.        cursorObject = connectionObject.cursor()
434.        sqlQuery = """CREATE TABLE IF NOT EXISTS Behaviour_Change_PCT (ID int(11) NOT NULL AUT
    O_INCREMENT, Timestamp_Start datetime NOT NULL, Key_id varchar(150) NOT NULL,
435.        User_id varchar(150) NOT NULL, Steps float(18) DEFAULT NULL, Sedentary_Duration float(
    18) DEFAULT NULL, Vigorous_Duration float(18) DEFAULT NULL,
436.        Social_Interaction float(18) DEFAULT NULL, Social_Isolation float(18) DEFAULT NULL, Ha
    ppiness_Duration float(18) DEFAULT NULL,
437.        Sadness_Duration float(18) DEFAULT NULL, Cognitive_Duration float(18) DEFAULT NULL, UN
    IQUE KEY ID (ID), PRIMARY KEY (Key_id)) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1"
    ""
438.
439.        # Execute the sqlQuery
440.        cursorObject.execute(sqlQuery)
441.        # SQL query string
442.        sqlQuery = "show tables"
443.        # Execute the sqlQuery
444.        cursorObject.execute(sqlQuery)
445.        #Fetch all the rows
446.        rows = cursorObject.fetchall()
447.
448.        for row in rows:
449.            print(row)
450.
451.    except Exception as e:
452.
453.        print("Exeception occured:{}".format(e))
454.
455.    finally:
456.
457.        connectionObject.close()
458.        cursorObject.close()
459.
460.    #Creates the Behaviour_Change_Quantifier Table
461.    connectionObject = mysql.connector.connect(host='HBAF_Server',user ='Short_Behaviour',pass
    wd='pswd',db='ShortTerm_Behaviour' )
462.    try:
463.        # Create a cursor object
464.        cursorObject = connectionObject.cursor()
465.        sqlQuery = """CREATE TABLE IF NOT EXISTS Behaviour_Change_Quantifier (ID int(11) NOT N
    ULL AUTO_INCREMENT, Key_id varchar(150) NOT NULL,
466.        User_id varchar(150) NOT NULL, Behaviour_Type varchar(150) DEFAULT NULL, Change_Num in
    t(11) DEFAULT NULL, Timestamp_Start datetime NOT NULL,
467.        Timestamp_End datetime NOT NULL, Total_Duration int(11), Mean float(18) DEFAULT NULL,
    Diff float(18) DEFAULT NULL,
468.        Relative_Diff float(18) DEFAULT NULL, UNIQUE KEY ID (ID), PRIMARY KEY (Key_id)) ENGINE
    =InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1"""
469.
470.        # Execute the sqlQuery
471.        cursorObject.execute(sqlQuery)
472.        # SQL query string
473.        sqlQuery = "show tables"
474.        # Execute the sqlQuery
475.        cursorObject.execute(sqlQuery)
476.        #Fetch all the rows
477.        rows = cursorObject.fetchall()
478.
479.        for row in rows:
480.            print(row)
481.
482.    except Exception as e:
483.
484.        print("Exeception occured:{}".format(e))
```

```
485.
486.    finally:
487.
488.        connectionObject.close()
489.        cursorObject.close()
```

5 Bibliography

Datatracker. (2018, December 20). *The JavaScript Object Notation (JSON) Data Interchange Format*. Retrieved from Datatracker: https://datatracker.ietf.org/doc/rfc8259/?include_text=1

ECMA. (2017, December 1). *The JSON Data Interchange Syntax - 2nd edition*. Retrieved from ecma-international: https://www.ecma-international.org/publications/standards/Ecma-404.htm

ENS Paris-Saclay, C. (2018, January 3). *Exact segmentation: dynamic programming*. Retrieved from Ruptures: https://ctruong.perso.math.cnrs.fr/ruptures-docs/build/html/detection/dynp.html

Framework Slim. (2017, November 4). *Slim 3.9.0 released*. Retrieved from Slim Framework: https://www.slimframework.com/2017/11/04/slim-3.9.0.html

Microsoft. (2014, May 5). *Microsoft Speech Platform SDK 11 Documentation*. Retrieved from Microsoft: https://docs.microsoft.com/en-us/previous-versions/office/developer/speech-technologies/dd266409(v%3doffice.14)

MySQL. (2019, November 26). *MySQL 5.7 Release Notes*. Retrieved from MySQL: https://dev.mysql.com/doc/relnotes/mysql/5.7/en/

Python. (2015, September 13). *Python 3.5.0*. Retrieved from Python: https://www.python.org/downloads/release/python-350/

Ubuntu. (2019, March 4). *Ubuntu 14.04.6 LTS (Trusty Tahr)*. Retrieved from Ubuntu Releases: http://releases.ubuntu.com/trusty/

Acknowledgements

The Council of Coaches project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Headings and titles in this document, as well as the Council of Coaches logo use the Comfortaa font, designed by Johan Aakerlund and Cyreal and licensed under the Open Font License[1].

Additional text in this document uses the Roboto font, designed by Christian Robertson and licensed under the Apache License, Version 2.0[2].

The Council of Coaches logo and Blobmen graphics were drawn freely in Inkscape, licensed under the GNU General Public License[3].

[1] Open Font License: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL_web

[2] Apache License, Version 2.0: http://www.apache.org/licenses/LICENSE-2.0

[3] Inkscape License Information: https://inkscape.org/about/license/