Пермский филиал федерального государственного автономного образовательного учреждения высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Образовательная программа бакалавриата «Бизнес-информатика»

**ОТЧЕТ**

**по производственной практике**

Выполнил студент БИ-17-2

Кочетов Денис Дмитриевич

(подпись)

Проверили:

Руководитель практики  
от предприятия

Разработчик Центра Цифровых Решений

Кротких Андрей Александрович \_\_\_\_\_\_\_\_\_\_\_

(10) (подпись)

(29.05.2020)

Руководитель практики  
 от НИУ ВШЭ – Пермь

Академический руководитель ОП: Бизнес – информатика

Шестакова Лидия Валентиновна

\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

(дата)

Пермь, 2020

Оглавление

[Введение 3](#_Toc56788323)

[Глава 1. Описание организации 4](#_Toc56788324)

[Описание IT инфраструктуры предприятия 6](#_Toc56788325)

[Глава 2. Описание индивидуального задания 8](#_Toc56788326)

[2.1 Предметная область 8](#_Toc56788327)

[2.2 Описание примененных технологий 10](#_Toc56788328)

[2.3. Описание данных 13](#_Toc56788329)

[2.4 Обработка данных 14](#_Toc56788330)

[Заключение 21](#_Toc56788331)

[Библиографический список 22](#_Toc56788332)

[Приложение А. Пример данных 23](#_Toc56788333)

[Приложение Б. Распределения значений 24](#_Toc56788334)

[Приложение В. Листинг 27](#_Toc56788335)

# Введение

Производственная практика проходила на базе компании ООО «Парма-Телеком» удаленно.

Целью практики было закрепить и развить полученные в рамках учебного курса навыки в ходе решения реальных производственных задач, приобрести навыки самостоятельной практической работы в составе корпоративной структуры. Планируемые результаты: разработка собственной гипотезы детектирования на размеченных данных.

Задачи практики:

1. Анализ предметной области индивидуального задания
2. Выбор информационных технологий для выполнения задания
3. Предобработка данных, анализ данных.
4. Разработка математических моделей

# Глава 1. Описание организации

Производственная практика проходила удаленно на базе организации ООО «Парма-Телеком». Организация была основана в городе Перми в 2004 году, в 2013 вошла в состав группы компаний ITPS (IT Professional Solutions). Парма-Телеком работает в области информационных технологий, консалтинга, ИТ- инжиниринга. Основной сферой деятельности является разработка цифровых решений и системная интеграция для крупных компаний в нефтегазовой, химической и металлургической областях.

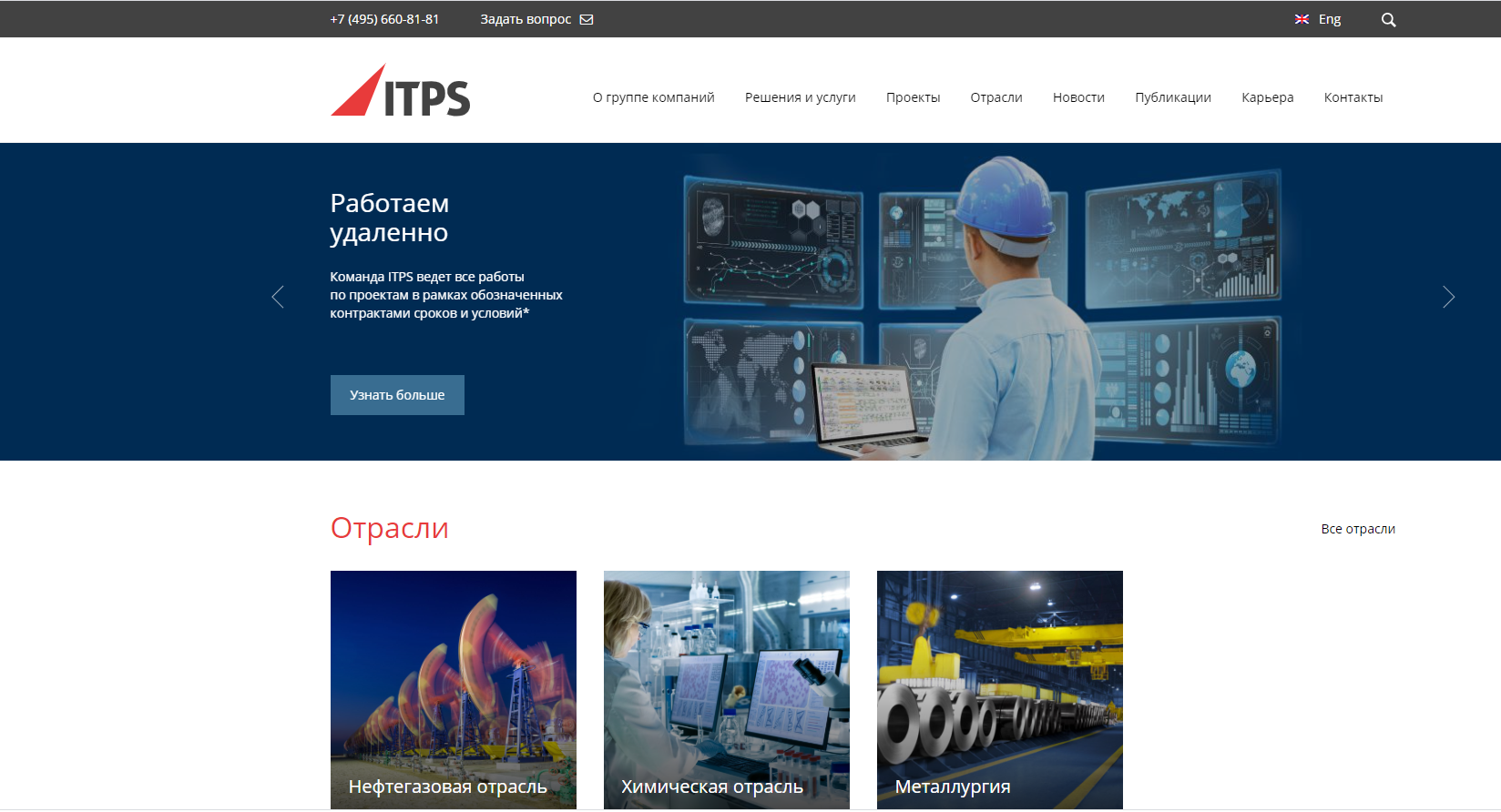


Рисунок 1.1. Главная страница сайта группы компаний ITPS

«Парма-Телеком» квалифицирована в качестве официального партнера и зарегистрированного поставщика услуг более чем 200 крупнейших разработчиков и поставщиков ИТ-решений для бизнеса.

Эксперты, разработчики и специалисты компании, обладают более чем 15-летним опытом разработки и внедрения комплексных информационных систем, уникальной экспертизой и компетенциями в современных цифровых технологиях. Компания разрабатывает и успешно внедряет комплексные решения для повышения эффективности производственных и технологических процессов на базе платформы собственной разработки – AVIST (свидетельство о государственной регистрации программ для ЭВМ №2013619085 от 25.09.2013), в частности, в нефтегазовом секторе.

Платформа и функциональные модули AVIST разрабатываются с использованием современных цифровых технологий, таких как промышленный интернет вещей (IIoT), сбор и первичная обработка данных реального времени (Big Data), цифровое моделирование, интеллектуальная обработка больших данных c реализацией умных правил, предиктивный анализ и оптимизация (Predictive Analysis&Optimization), возможности машинного обучения.

Рисунок 1.2. Страница сайта ITPS с описанием платформы AVIST

В активе компании:

Реализация ИТ-проектов от разработки концепции до практического внедрения и передачи в промышленную эксплуатацию.

Опыт создания систем управления активами и проектами с годовым бюджетом более $2 млрд в год.

Поддержка 30 000 конечных пользователей, ежедневно работающих во внедренных системах.

Проектирование, строительство, сдача в эксплуатацию и дальнейшее сопровождение производственных объектов и систем АСУ ТП для предприятий нефтегазодобывающей отрасли в России и странах СНГ.

В штате всей группы компаний ITPS – больше 400 сотрудников. География деятельности – Россия, Восточная и Западная Европа, Латинская Америка, Северная Африка, Ближний и Средний Восток.

Описание IT инфраструктуры предприятия

Компания пользуется различными корпоративными сервисами для коммуникаций между сотрудниками и поддержанием разработок проектов.

Коммуникация внутри компании происходит с помощью корпоративной электронной почты через Microsoft Outlook, а также при помощи коммуникационной программы-клиента Skype for Business.

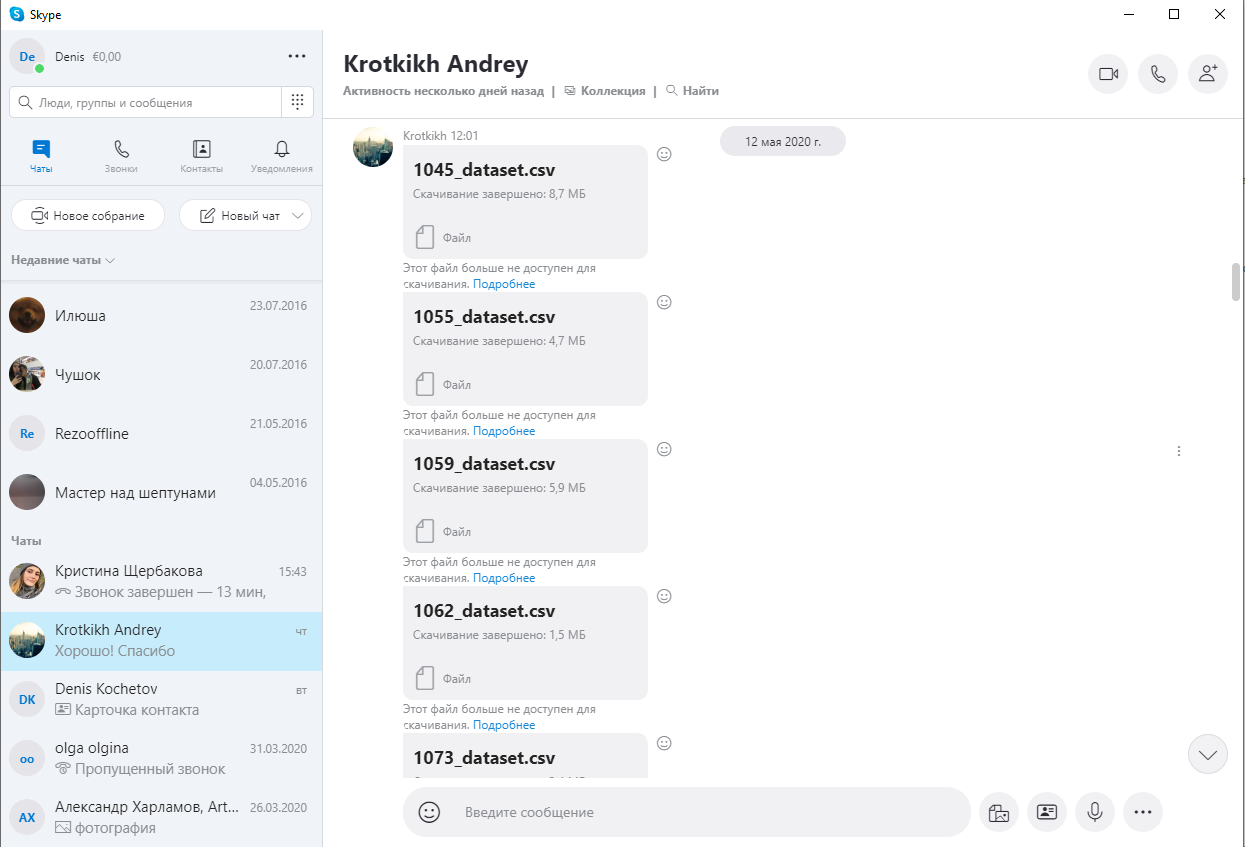
В масштабе производственной практики мне не были выданы аккаунты в корпоративных сервисах, коммуникации проходили через личную почту и аккаунт в Skype.

Рисунок 2.1 Коммуникация в Skype

Такой формат взаимодействия позволяет легко влиться в систему коммуникаций компании, так как времени на регистрацию в сервисах и на освоение работы с ними не требуется, благодаря их простоте, доступности и распространенности.

# Глава 2. Описание индивидуального задания

На практику были получены следующие задания:

1. Предобработка данных.
2. Анализ данных.
3. Разработка математических моделей способных детектировать аномальные участки данных, в том числе и в предиктиве.

Данные представляют собой записи о состоянии человека, приходящие с его фитнес-браслета каждую минуту. Состояние определяется по таким параметрам как пульс, гидратация, количество пройденных шагов и другие.

2.1 Предметная область

Представленные данные отражали различные биологические данные человека, носящего фитнес-браслет, приходящие в систему каждую минуту. По эти данным необходимо было сделать выводы о связи различных показателей и физиологического состояния человека.

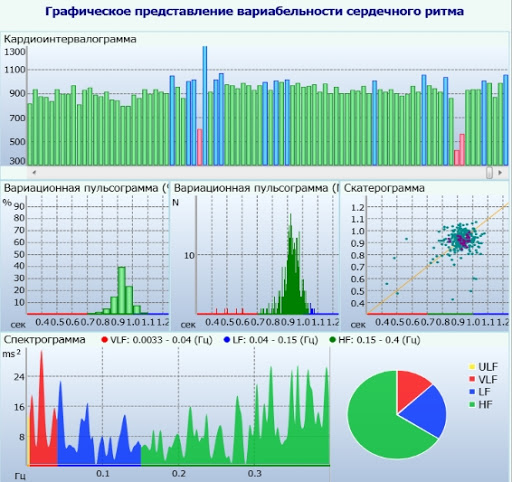
Выводы о физиологическом состоянии человека на основании данных о его сердечном ритме. В медицине существует показатель – вариабельность сердечного ритма – это изменение сердечного ритма человека в интервале времени от начала цикла одного сердечного сокращения до начала другого. Оно наблюдается даже в состояния покоя, то есть во сне [1].

Рисунок 2.1. График вариабельности сердечного ритма

В норме частота пульса у взрослого человека в покое 50-85 ударов в минуту. При нагрузке, при занятиях спортом пульс может значительно повышаться. Людям с заболеваниями сердечно сосудистой системы рекомендуется регулярно измерять пульс при физической нагрузке. Не рекомендуется, чтобы она превышала 85% от максимальной ЧСС. Максимальная же ЧСС рассчитывается по формуле: 220-возраст. К примеру, если человеку 40 лет, то максимальная ЧСС составит: 220-40=180 ударов в минуту. Следует помнить, что у детей частота пульса и ЧСС выше, чем у взрослых.

Во-первых, пульс стоит измерять в состоянии покоя. Если он меньше 55 или более 85 в покое, то следует посетить кардиолога. Подобные отклонения не всегда говорят о патологии. К примеру, у спортсменов пульс обычно ниже, чем у обычных людей. Для спортсмена пульс в покое 40 ударов в минуту может быть нормой. Так что, если человек занимается спортом, даже на любительском уровне, пульс в 50 ударов в минуту может быть нормой. Если же человек не спортивный, то подобный пульс может привести к головокружению и быть симптомом заболевания. В любом случае, чтобы исключить возможные заболевания, при отклонениях от вышеописанных норм, следует посетить кардиолога.  
Также, если при измерении пульса вы чувствуете выраженную неравномерность, нерегулярность пульса, также обратитесь к врачу. Подобная нерегулярность пульса может быть связана с экстрасистолией, то есть внеочередными сокращениями сердца, либо фибрилляцией предсердий.

По данным с браслетов рассчитать вариабельность сердечного ритма не представляется возможным, для этого необходимы дополнительные источники данных (например электрокардиограмма). Руководителем практики было предложено рассмотреть имеющиеся показатели в различные периоды жизнедеятельности, придумать и рассчитать различные метрики, найти закономерности и сопоставить с известными данными о болезнях или недомоганиях владельцев браслетов.

Общий подход определения рисков для здоровья базируется на концепции промышленного интернета вещей и систем. Его использование обусловлено появлением возможности дистанционного получения, накопления, обработки и анализа больших, в том числе неструктурированных, данных с браслета в режиме, близком к реальному времени.

При этом подходе оценка рисков для здоровья человека по психофизиологическим параметрам, измеряемым с помощью браслета, заключается в обработке и преобразовании измеренных значений для последующего сопоставления с заранее установленными шаблонами состояний здоровья, которые можно определить по этим измерениям, и в оценивании текущего состояния в отношении возможных рисков для здоровья в режиме, близком к реальному времени.

Получаемые значения параметров, отражающих изменения и динамику психофизиологического состояния человека и, в частности, уровня стресса, в совокупности представляют собой основу для широкого спектра выводов в вероятностной форме. Каждый из выводов основывается на результате объективного наблюдения за состоянием здоровья, состоящего из учета значимых с медицинской точки зрения закономерностей изменения значений параметров.

Из составленного на основе общепринятых медицинских знаний банка наблюдений за параметрами, предоставляемыми браслетом, могут быть сформированы шаблоны по значениям параметров, характеризующих предположения о различных рисках для здоровья человека, включая и гипотезу о вероятности нахождения в практически здоровом состоянии.

Таким образом, основная идея подхода состоит в сопоставлении текущего набора значений наблюдаемых параметров с имеющимися шаблонами, что позволяет выдвигать обоснованные гипотезы о рисках для здоровья наблюдаемого.

2.2 Описание примененных технологий

Все действия с данными проводились с помощью языка программирования Python, который очень распространен среди специалистов по работе с данными. В нем можно найти большое количество библиотек, документации, примеров и советов по работе с анализом данных. Средой разработки был выбран PyCharm, так как именно его предлагают для работы с языком Python на курсах от GeekBrains, которые были пройдены ранее.

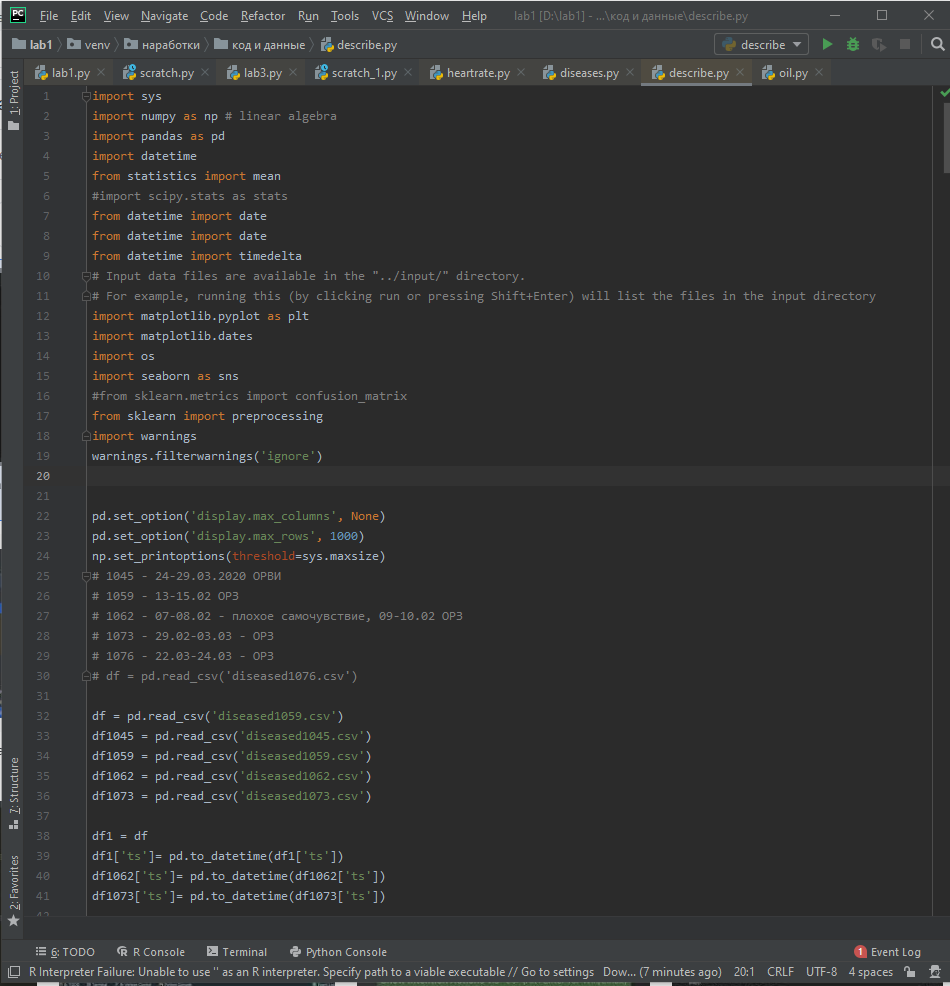


Рисунок 2.2. Интерфейс PyCharm

Для работы с данными были использованы библиотеки pandas, numpy, statistics, matplotlib, scikit-learn.

Библиотека Pandas содержит огромное количество функций для работы с двумерными массивами (dataframe), которыми являются таблицы с данными.

 Рисунок 2.2. Pandas логотип

Matplotlib была использована для визуализации данных в виде различных графиков.

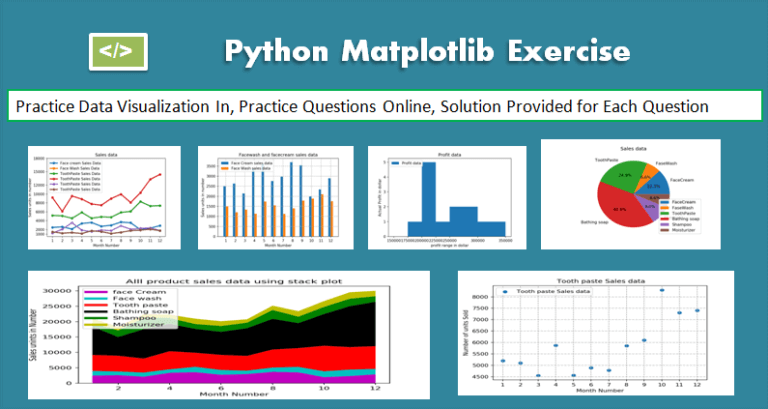


Рисунок 2.3. Matplotlib, примеры графиков

С помощью функций библиотеки scikit-learn данные были нормализованы для дальнейшей работы с ними.

Numpy помогла быстро и легко проводить различные математические операции над данными.

При помощи библиотеки statistics удобно считать статистические показатели разнообразных форматов данных.

2.3. Описание данных

Для работы были представлены файлы csv шести владельцев браслетов, по пяти из них также дополнительно была известна информация об их самочувствии в некоторые периоды пользования девайсом.

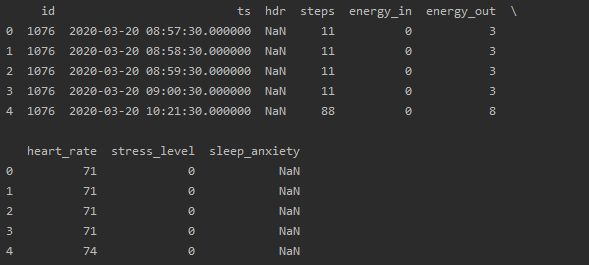
Данные представлены в виде таблиц в формате csv.

Рисунок 2.1. Пример данных клиента 1076

Описание структур данных представлено на таблице ниже.

Таблица 2.1. Описание структур данных

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Показатель** | **Формат** |
| id | номер клиента | int64 |
| timestamp | время фиксации данных | datetime |
| hdr | гидратация (обеспеченность организма жидкостью) | float64 |
| steps | количество шагов | int64 |
| energy\_in | получено калорий | int64 |
| energy\_out | потрачено калорий | int64 |
| heart\_rate | сердечный ритм | int64 |
| stress\_level | уровень стресса | int64 |
| sleep\_anxiety | тревожность сна | float64 |

2.4 Обработка данных

Перед тем, как строить какие-либо выводы о представленных данных, необходимо их обработать.

Задание 1. Выделить периоды снов. Сгруппировать данные по периодам снов для каждого человека (между последним timestamp первого и первым timestamp второго сна не меньше получаса, если больше, то два разных сна)

df['ts']= pd.to\_datetime(df['ts']) #привести столбец даты к выбранному формату  
  
df['sleep\_number'] = 0  
df['week\_number'] = 0  
  
df['sleep\_number'] = np.where(np.isnan(df['sleep\_anxiety']), 0, 1) #столбец номер сна  
  
df1 = df  
df2 = df1[df1.sleep\_number != 0].reset\_index()  
  
j = 1  
for i in range(len(df2)-1):  
 if (df2.loc[i + 1, "ts"] - df2.loc[i, "ts"]).total\_seconds()/60 > 30:  
 j += 1  
 df2.loc[i + 1, "sleep\_number"] = j  
   
j = 1  
timestamp = df2.loc[0, "ts"]  
for i in range(len(df2)):  
 if (df2.loc[i, "ts"] - timestamp).total\_seconds()/60/60/24 >= 7 :  
 timestamp = df2.loc[i, "ts"]  
 j += 1  
 df2.loc[i, "week\_number"] = j  
  
print(df2)  
  
df2.to\_csv ('sleep\_number1076.csv', index = False, header=True) #записать данные в файл

Задание 2. Отразить в данных промежутки болезни

# 1045 - 24-29.03.2020 ОРВИ  
# 1059 - 13-15.02 ОРЗ  
# 1062 - 07-08.02 - плохое самочувствие, 09-10.02 ОРЗ  
# 1073 - 29.02-03.03 - ОРЗ  
# 1076 - 22.03-24.03 - ОРЗ  
startDate = datetime.datetime(2020, 2, 29) #зафиксировать стартовую дату  
endDate = datetime.datetime(2020, 3, 3) #зафиксировать конечную дату  
#startDate1 = datetime.datetime(2020, 2, 9) #зафиксировать стартовую дату  
#endDate1 = datetime.datetime(2020, 2, 10) #зафиксировать конечную дату  
  
df['diseased'] = np.where((df['ts'] <= endDate) & (df['ts'] >= startDate), 1, 0) #переменная   
df1 = df[df.diseased != 0]  
print(df)  
df1.ts = pd.to\_datetime(df.ts)  
df1.set\_index('ts', inplace=True)  
df.to\_csv ('diseased1073.csv', index = False, header=True) #записать данные в файл

Задание 3. Посмотреть различные статистические показатели данных

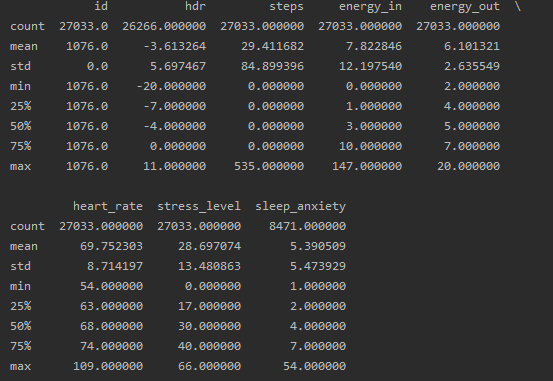


Рисунок 2.2. Пример статистических показателей данных

Задание 4. Отметить на различных графиках время болезни клиента.

Оранжевым цветом отмечены дни болезни клиента.

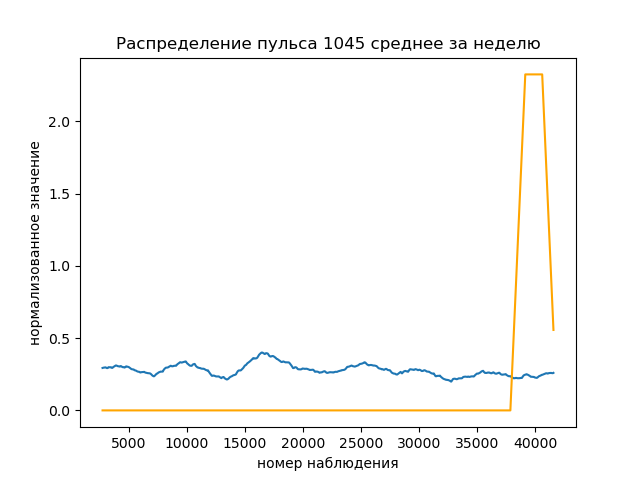


Рисунок 2.3. График скользящего среднего значения пульса клиента 1045 за неделю

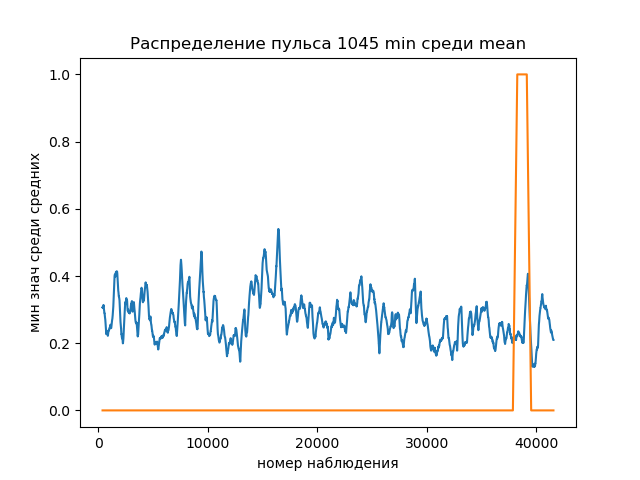


Рисунок 2.4. График, показывающий минимальное значения среди средних для клиента 1045 за неделю

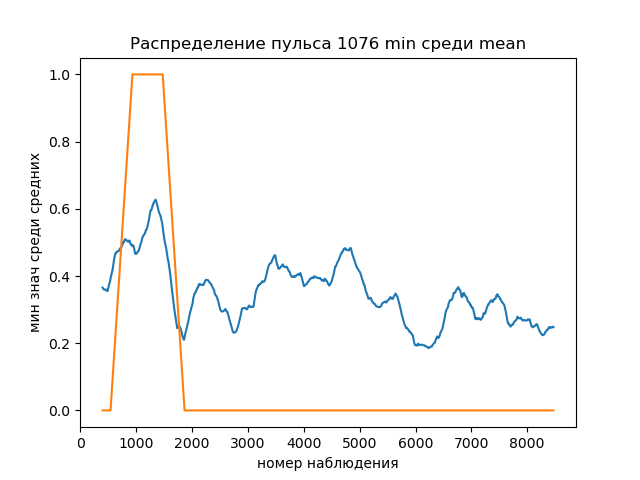


Рисунок 2.5. График, показывающий минимальное значения среди средних для клиента 1076 за неделю

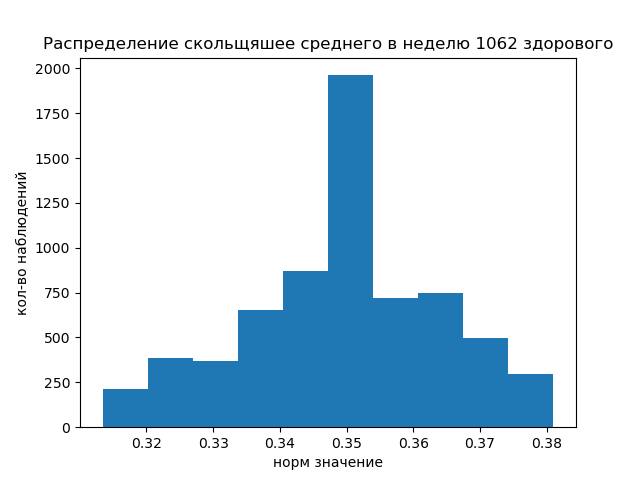


Рисунок 2.6. Гистограмма распределение скользящего среднего нормализованного значения пульса на неделю

Задание 4. Посчитать конечные разности между значениями и отразить на графике

#возвращает список модулей разностей между элементами массива  
def difference(array):  
 df = []  
 for i in range (len(array)-1):  
 df.append(abs(array[i+1] - array[i]))  
 df.append(array[i+1] - array[i])  
 return df  
#возвращает дф разностей между элементами столбцов датафрейма  
def different(df\_scaled):  
 df\_dif = pd.DataFrame()  
 df\_dif['heart\_rate'] = difference(df\_scaled.heart\_rate)  
 df\_dif['hdr'] = difference(df\_scaled.hdr)  
 df\_dif['stress\_level'] = difference(df\_scaled.stress\_level)  
 df\_dif['sleep\_anxiety'] = difference(df\_scaled.sleep\_anxiety)  
 df\_dif['diseased'] = difference(df\_scaled.diseased)  
 return df\_dif

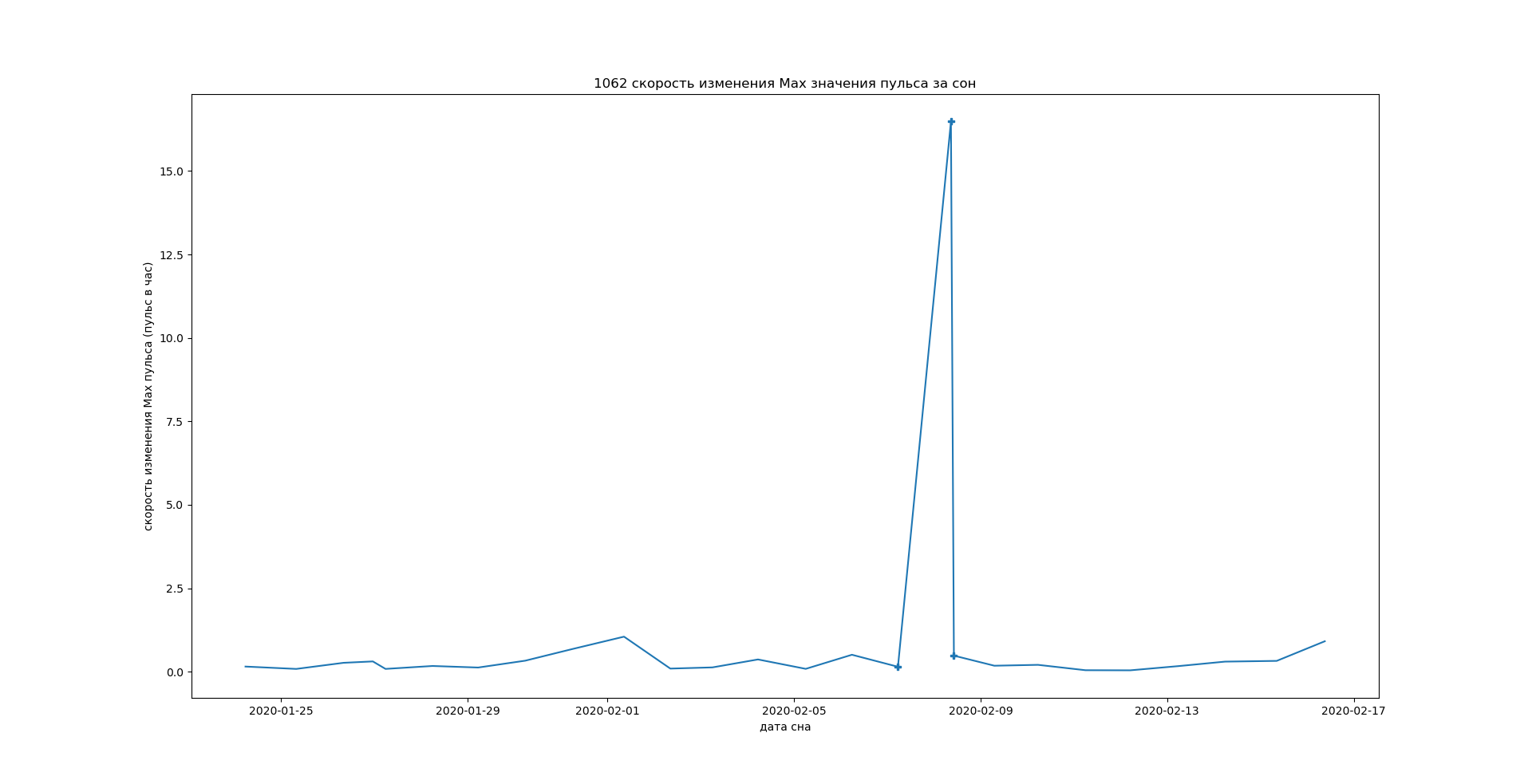
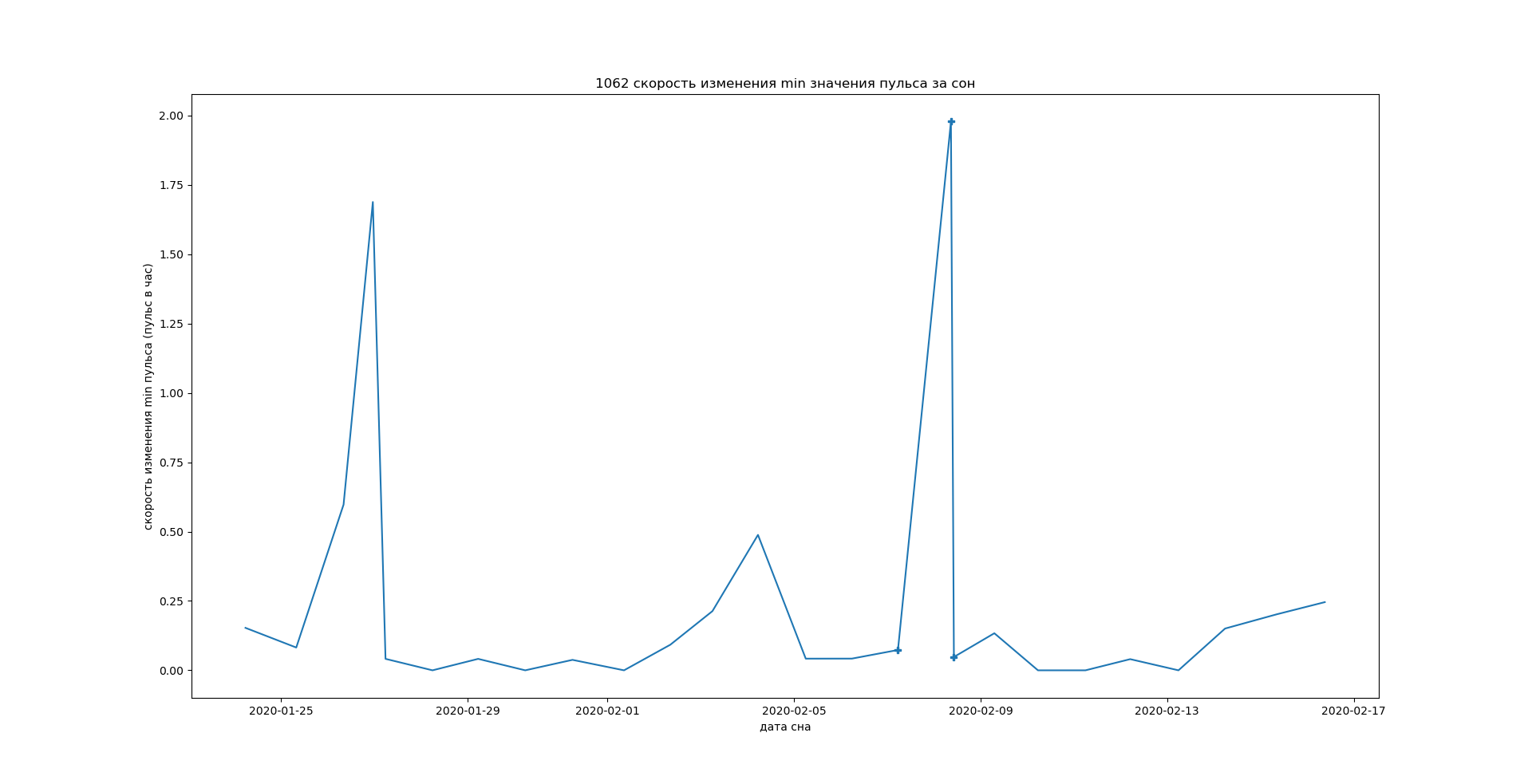
  
Рисунок 2.7. Изменение максимального пульса 1062 за сон

Рисунок 2.8. Изменение минимального пульса 1062 за сон

Задание 5. Выделить интервалы активности

stepList = [] #временные интервалы шагов  
stepDays = [] #дни когда шаги шагались  
dayStep = [] #заносит ts интервала шагов в конце обнуляется  
for i in range (len(dfSteps)-1): #гиги за шаги  
 dayStep.append(dfSteps.loc[i, 'ts'])  
 if (dfSteps.loc[i + 1, 'ts'] - dfSteps.loc[i, 'ts']).total\_seconds() / 60 > 10:  
 interval = (dayStep[-1] - dayStep[0]).total\_seconds()/60  
 stepList.append(interval)  
 stepDays.append(dfSteps.loc[i, 'day'])  
 dayStep = []

Рисунок 2.9. Средние интервалы активности для клиента 1059

Задание 6. Написать функцию, детектирующую отклонения от нормальных данных

#возвращает список дат когда отклонение было больше определенного  
def checkdate(dfx, val):  
 df = []  
 for i in range (len(dfx)):  
 if abs(dfx.heart\_rate[i]) > val:  
 df.append(dfx.ts[i])  
 return df

Рисунок 2.10. Вывод дат с отклонением среднего пульса

# Заключение

В результате прохождения производственной практики на базе компании Парма-Телеком были получены практические навыки работы с данными, их обработкой, анализом и построением гипотез на их основе.

Были изучены теоретические материалы по связи сердечного ритма человека и его физического состояния.

По окончанию практики были получены знания относительно работы в компании разработчике программного обеспечения для нефтегазовой, химической и металлургической отраслей. В частности, были получены знания о таком направлении как анализ данных. Были изучены новые инструменты и технологии. Получены новые знания и навыки для дальнейшей карьеры в сфере анализа данных.

Реализована возможность работы под руководством опытного специалиста, получение необходимого опыта передачи программистской мудрости.

Все поставленные задачи были выполнены, цели достигнуты.

# Библиографический список

1. Бокерия Л.А., Бокерия О.Л., Волковская И.В. – Вариабельность сердечного ритма: методы измерения, интерпретация, клиническое использование// Анналы Аритмологии – 2009 – №4.
2. Баевский Р.М., Иванов Г.Г. - Вариабельность сердечного ритма: теоретические аспекты и возможности клинического применения // Ультразвуковая и функциональная диагностика - 2001 - №3.
3. Документация Python [Электронный ресурс] Режим доступа: <https://www.python.org/doc/>, свободный.
4. Документация Pandas [Электронный ресурс] Режим доступа: <https://pandas.pydata.org/docs/>, свободный
5. Документация Scikit-learn [Электронный ресурс] Режим доступа: <https://scikit-learn.org/stable/user_guide.html>, свободный
6. Документация Matplotlib [Электронный ресурс] Режим доступа: <https://matplotlib.org/contents.html>, свободный
7. Сайт группы компаний ITPS [Электронный ресурс] Режим доступа: <http://itps.com/>, свободный

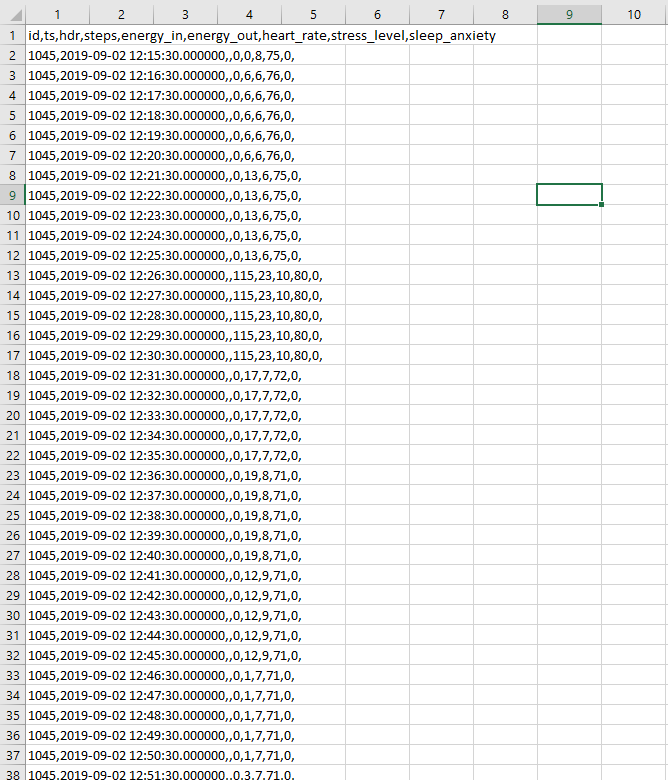
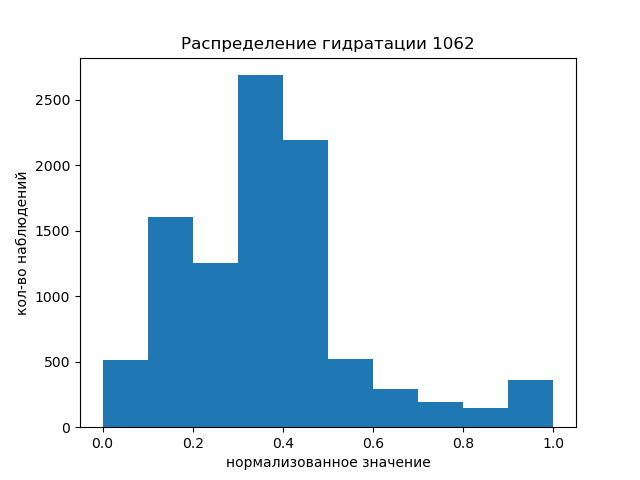
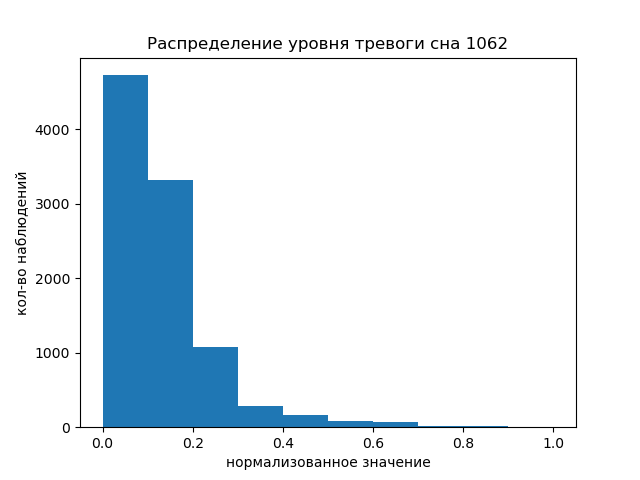
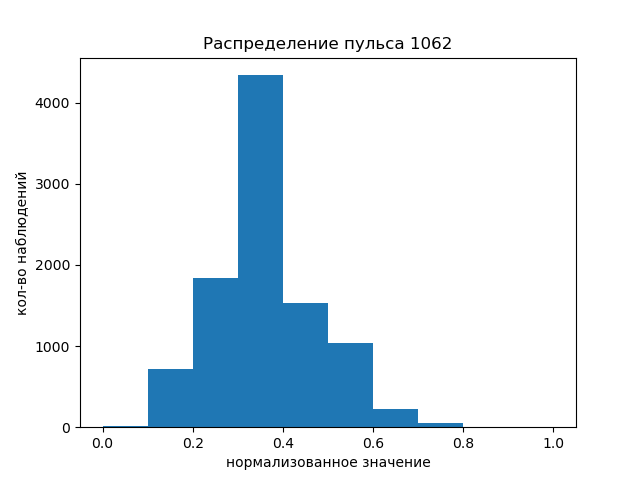
Приложение А. Пример данных

Рисунок А.1. Данные по клиенту 1045

# Приложение Б. Распределения значений



# Приложение В. Листинг

import numpy as np  
import pandas as pd  
import datetime as dt  
from datetime import date  
from datetime import time  
from datetime import datetime  
import matplotlib.pyplot as plt  
import os  
import seaborn as sns  
import datetime  
from datetime import date  
from datetime import date  
from datetime import timedelta  
import warnings  
warnings.filterwarnings('ignore')  
#dfName = '1062.csv'  
#dfName = '1045.csv'  
dfName = '1059.csv'  
#dfName = '1073.csv'  
# dfName = '1076.csv'  
df = pd.read\_csv(dfName)  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.max\_rows', 100)  
#print(df)  
  
# print(dataset.shape)  
#dateFormat = '%Y-%m-%d' #определить формат даты  
  
df['day'] = 0  
df['ts']= pd.to\_datetime(df['ts']) #привести столбец даты к выбранному формату  
df['day'] = df['ts'].dt.date  
  
  
df['sleep\_number'] = 0  
df['week\_number'] = 0  
  
# for i in range (20000):  
# df.loc[i, 'day'] = df.loc[i, 'ts'].strftime("%d")  
  
df['sleep\_number'] = np.where(np.isnan(df['sleep\_anxiety']), 0, 1) #столбец номер сна  
  
df1 = df  
dfSteps = df1.drop(columns=['id', 'energy\_in', 'energy\_out', 'hdr', 'stress\_level', 'heart\_rate', 'sleep\_anxiety'])  
dfSteps = dfSteps[dfSteps.steps != 0].reset\_index() #остались лишь шаги детка  
  
print('Интервалы шагов')  
print(dfSteps[['ts','day', 'steps']][:300])  
min15 = []  
ts15 = []  
chill = []  
for i in range (len (df1) - 1): #15 МИНУТ ДО СНА минимальный пульс  
 if (df1.loc[i+1, 'sleep\_number'] > df1.loc[i, 'sleep\_number'] ):  
 for j in range (15):  
 chill.append(df1.loc[i-j, 'heart\_rate'])  
 min15.append(min(chill))  
 ts15.append(df1.loc[i, 'ts'].strftime("%m %d"))  
 chill = []  
#print(min15)  
stepList = [] #временные интервалы шагов  
stepDays = [] #дни когда шаги шагались  
dayStep = [] #заносит ts интервала шагов в конце обнуляется  
for i in range (len(dfSteps)-1): #гиги за шаги  
 dayStep.append(dfSteps.loc[i, 'ts'])  
 if (dfSteps.loc[i + 1, 'ts'] - dfSteps.loc[i, 'ts']).total\_seconds() / 60 > 10:  
 interval = (dayStep[-1] - dayStep[0]).total\_seconds()/60  
 stepList.append(interval)  
 stepDays.append(dfSteps.loc[i, 'day'])  
 dayStep = []  
dfStepDays = pd.DataFrame()  
dfStepDays['interval'] = stepList  
dfStepDays['interval'] = dfStepDays['interval'].astype(float)  
dfStepDays['day'] = stepDays  
print(dfStepDays)  
# stepDays = [] #дни когда шаги шагались  
# stepList = [] #временные интервалы шагов  
  
activityDays = [] #дни когда шаги шагались  
activityTime = [] #временные интервалы шагов  
tempStep = []  
for i in range (len(dfStepDays)-1):  
 tempStep.append(dfStepDays.loc[i, 'interval'])  
 if (dfStepDays.loc[i, 'day'] != dfStepDays.loc[i+1, 'day']):  
 activityDays.append(dfStepDays.loc[i, 'day'])  
  
 activityTime.append(sum(tempStep) / len(tempStep))  
 tempStep = []  
  
print(activityDays)  
print(activityTime)  
  
  
df2 = df1[df1.sleep\_number != 0].reset\_index() #произошел дроп дней, остались лишь сны..  
  
j = 1  
for i in range(len(df2)-1):  
 if (df2.loc[i + 1, "ts"] - df2.loc[i, "ts"]).total\_seconds()/60 > 30:  
 j += 1  
 df2.loc[i + 1, "sleep\_number"] = j  
  
j = 1  
timestamp = df2.loc[0, "ts"]  
for i in range(len(df2)):  
 if (df2.loc[i, "ts"] - timestamp).total\_seconds()/60/60/24 >= 7 :  
 timestamp = df2.loc[i, "ts"]  
 j += 1  
 df2.loc[i, "week\_number"] = j  
  
print(df2)  
fileName = 'sleep\_number' + dfName  
#print(fileName)  
  
df2.to\_csv (fileName, index = False, header=True) #записать данные в файл  
  
  
  
####ОТМЕТИТЬ ДНИ БОЛЕЗНЕЙ#############  
  
#dfName = 'sleep\_number1062.csv'  
#dfName = 'sleep\_number1045.csv'  
dfName = 'sleep\_number1059.csv'  
#dfName = 'sleep\_number1073.csv'  
# dfName = 'sleep\_number1076.csv'  
df = pd.read\_csv(dfName)  
from collections import Counter  
check = Counter(df.sleep\_number).values()  
ch = np.sum(list(check))  
print(check)  
print(ch)  
print(df.sleep\_number.unique())  
df['ts']= pd.to\_datetime(df['ts']) #привести столбец даты к выбранному формату  
  
# 1045 - 24-29.03.2020 ОРВИ  
# 1059 - 13-15.02 ОРЗ  
# 1062 - 07-08.02 - плохое самочувствие, 09-10.02 ОРЗ  
# 1073 - 29.02-03.03 - ОРЗ  
# 1076 - 22.03-24.03 - ОРЗ  
if dfName == 'sleep\_number1062.csv':  
 startDate = datetime.datetime(2020, 2, 7) # зафиксировать стартовую дату  
 endDate = datetime.datetime(2020, 2, 10) # зафиксировать конечную дату  
 fileName = 'diseased1062.csv'  
elif dfName == 'sleep\_number1045.csv':  
 startDate = datetime.datetime(2020, 3, 24) # зафиксировать стартовую дату  
 endDate = datetime.datetime(2020, 3, 29) # зафиксировать конечную дату  
 fileName = 'diseased1045.csv'  
elif dfName == 'sleep\_number1059.csv':  
 startDate = datetime.datetime(2020, 2, 13) # зафиксировать стартовую дату  
 endDate = datetime.datetime(2020, 2, 15) # зафиксировать конечную дату  
 fileName = 'diseased1059.csv'  
 #dis1059 = ['2020-02-13', '2020-02-14', '2020-02-15']  
elif dfName == 'sleep\_number1073.csv':  
 startDate = datetime.datetime(2020, 2, 29) # зафиксировать стартовую дату  
 endDate = datetime.datetime(2020, 3, 3) # зафиксировать конечную дату  
 fileName = 'diseased1073.csv'  
elif dfName == 'sleep\_number1076.csv':  
 startDate = datetime.datetime(2020, 3, 22) # зафиксировать стартовую дату  
 endDate = datetime.datetime(2020, 3, 24) # зафиксировать конечную дату  
 fileName = 'diseased1076.csv'  
  
#startDate = datetime.datetime(2020, 2, 7) #зафиксировать стартовую дату  
#endDate = datetime.datetime(2020, 2, 10) #зафиксировать конечную дату  
#startDate1 = datetime.datetime(2020, 2, 9) #зафиксировать стартовую дату  
#endDate1 = datetime.datetime(2020, 2, 10) #зафиксировать конечную дату  
  
df['diseased'] = np.where((df['ts'] <= endDate) & (df['ts'] >= startDate), 1, 0)  
#df['diseased'] = np.where((df['ts'] <= endDate1) & (df['ts'] >= startDate1), 1.5, df['diseased'])  
  
df1 = df[df.diseased != 0]  
print(df)  
df1.ts = pd.to\_datetime(df.ts)  
df1.set\_index('ts', inplace=True)  
df.to\_csv (fileName, index = False, header=True) #записать данные в файл  
  
  
#df = df.drop(columns=['index', 'id', 'sleep\_number', 'steps' ])  
#df1 = df1.drop(columns=['index', 'id', 'sleep\_number', 'steps' ])  
print(df.describe())  
print(df1.describe())  
#print (df.skew())  
#print (df1.skew())  
#print(df.corr())  
#print(df1.corr())  
print('КОРРЕЛЯЦИЯ')  
print(df.corr(method='spearman'))  
print(df1.corr(method='spearman'))  
  
print(df)  
print(df1)  
  
dfchill = pd.DataFrame()  
dfchill['ts'] = ts15  
dfchill['hr'] = min15  
chillname = 'chill' + fileName  
dfchill.to\_csv(chillname, index=False, header=True)  
plt.figure(0)  
x\_ticks = np.arange(0, 100, 7)  
plt.xticks(x\_ticks)  
plt.title('Минимальный пульс за 15 минут до сна ' + dfName)  
markers1059 = []  
#даты в новом представлении  
#print(ts15)  
markers1059.append(ts15.index('02 13'))  
markers1059.append(ts15.index('02 14'))  
#markers1059.append(ts15.index('02 15'))  
  
  
plt.plot(ts15, min15, 'o-', label = '15', markevery = markers1059)  
plt.figure(1)  
plt.title('Средние интервалы активности по датам ' + fileName)  
dis1059 = []  
  
#dis1059 = ['2020-02-13', '2020-02-14', '2020-02-15']  
print('Даты активности')  
print(activityDays)  
#dis1059 = [[datetime.date(2020, 2, 13), datetime.date(2020, 2, 14), datetime.date(2020, 2, 15)]]  
# dis1059.append(activityDays.index('2020-02-13'))  
dis1059.append(activityDays.index(datetime.date(2020, 2, 13)))  
dis1059.append(activityDays.index(datetime.date(2020, 2, 14)))  
dis1059.append(activityDays.index(datetime.date(2020, 2, 15)))  
# dis1059.append(activityDays.index('2020-02-15'))  
plt.plot(activityDays, activityTime, 'o--', markevery = dis1059)  
plt.show()

import sys  
import numpy as np # linear algebra  
import pandas as pd  
import datetime  
from statistics import mean  
#import scipy.stats as stats  
from datetime import date  
from datetime import date  
from datetime import timedelta  
# Input data files are available in the "../input/" directory.  
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory  
import matplotlib.pyplot as plt  
import matplotlib.dates  
import os  
import seaborn as sns  
#from sklearn.metrics import confusion\_matrix  
from sklearn import preprocessing  
import warnings  
warnings.filterwarnings('ignore')  
  
  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.max\_rows', 1000)  
np.set\_printoptions(threshold=sys.maxsize)  
# 1045 - 24-29.03.2020 ОРВИ  
# 1059 - 13-15.02 ОРЗ  
# 1062 - 07-08.02 - плохое самочувствие, 09-10.02 ОРЗ  
# 1073 - 29.02-03.03 - ОРЗ  
# 1076 - 22.03-24.03 - ОРЗ  
# df = pd.read\_csv('diseased1076.csv')  
  
df = pd.read\_csv('diseased1059.csv')  
df1045 = pd.read\_csv('diseased1045.csv')  
df1059 = pd.read\_csv('diseased1059.csv')  
df1062 = pd.read\_csv('diseased1062.csv')  
df1073 = pd.read\_csv('diseased1073.csv')  
  
df1 = df  
df1['ts']= pd.to\_datetime(df1['ts'])  
df1062['ts']= pd.to\_datetime(df1062['ts'])  
df1073['ts']= pd.to\_datetime(df1073['ts'])  
  
dateFormat = '%Y-%m-%d' #определить формат даты  
df1.ts = pd.to\_datetime(df1.ts, format = dateFormat) #привести столбец даты к выбранному формату  
#df1.ts = matplotlib.dates.date2num(df1.ts)  
df1.set\_index('ts')  
#print(df1.ts)  
  
#выводит значения heartrate в любой выбранный сон#  
def dreams(df1, num):  
 list\_dream = []  
 for i in range (len(df1)):  
 if df1.sleep\_number[i] == num:  
 list\_dream.append(df1.heart\_rate[i])  
 return list\_dream  
#выводит значения heartrate в любую выбранную неделю  
def weeks(df1, num):  
 list\_dream = []  
 for i in range (len(df1)):  
 if df1.week\_number[i] == num:  
 list\_dream.append(df1.heart\_rate[i])  
 return list\_dream  
  
list1 = dreams(df1, 1)  
list2 = weeks(df1, 2)  
# print('hr в снах')  
# print(list1)  
  
  
df = df.drop(columns=['index', 'id','steps', 'energy\_in', 'energy\_out', 'ts', 'day' ])  
#df1062 = df1062.drop(columns = ['index', 'id', 'steps', 'energy\_in', 'energy\_out', 'ts'])  
df = df.dropna()  
  
print('исходные данные')  
print(df.describe())  
print(df.head())  
  
#df = df[df.diseased == 0]  
#print(df1.describe())  
  
#df = df.drop(columns=['diseased'])  
  
min\_max\_scaler = preprocessing.MinMaxScaler()  
df\_minmax = min\_max\_scaler.fit\_transform(df)  
#minmax1062 = min\_max\_scaler.fit\_transform(df1062)  
# X\_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))  
# X\_scaled = X\_std \* (max - min) + min  
  
#нормализация данных через стандартное отклонение  
def normalize(x):  
 min\_max\_scaler = preprocessing.MinMaxScaler()  
 scaled1 = pd.DataFrame(min\_max\_scaler.fit\_transform(x), columns=x.columns)  
 return scaled1  
#df1026\_scaled = normalize(df1062, minmax1062)  
df\_scaled = pd.DataFrame(min\_max\_scaler.fit\_transform(df),columns = df.columns)  
#df1\_scaled = pd.DataFrame(min\_max\_scaler.fit\_transform(df1),columns = df1.columns)  
#df1\_scaled = normalize(df1)  
#print('нормализованные данные')  
#print(df\_scaled.describe())  
#normalized = preprocessing.normalize(df)  
# print('normalized')  
# print(normalized[:200])  
  
week = 2739 #длина недели  
night = 391 #длина сна  
dfMA = pd.DataFrame()  
  
# возвращает дф со скользящими средними значениями за выбранный период  
def window(df3, period):  
 dfMA = pd.DataFrame()  
 dfMA['heart\_rate'] = df3['heart\_rate'].rolling(window=period).mean()  
 dfMA['hdr'] = df3['hdr'].rolling(window=period).mean()  
 dfMA['stress\_level'] = df3['stress\_level'].rolling(window=period).mean()  
 dfMA['sleep\_anxiety'] = df3['sleep\_anxiety'].rolling(window=period).mean()  
 dfMA['diseased'] = df3['diseased'].rolling(window=period).mean()  
 dfMA = dfMA.dropna()  
 return dfMA  
  
# возвращает дф с минимальными значениями за выбранный период  
def window\_min(df3, period):  
 dfMA = pd.DataFrame()  
 dfMA['heart\_rate'] = df3['heart\_rate'].rolling(window=period).min()  
 dfMA['hdr'] = df3['hdr'].rolling(window=period).min()  
 dfMA['stress\_level'] = df3['stress\_level'].rolling(window=period).min()  
 dfMA['sleep\_anxiety'] = df3['sleep\_anxiety'].rolling(window=period).min()  
 dfMA['diseased'] = df3['diseased'].rolling(window=period).min()  
 dfMA = dfMA.dropna()  
 return dfMA  
  
#возвращает mean min max пульс за сон  
def checkDream(df1):  
 dreamList = []  
 minList = []  
 maxList = []  
 meanList =[]  
 tsList = []  
 disList = []  
 for i in range (len(df1) - 1):  
 dreamList.append(df1.loc[i, 'heart\_rate'])  
 if (df1.loc[i+1, 'sleep\_number'] != df1.loc[i, 'sleep\_number']):  
 minList.append(min(dreamList))  
 maxList.append(max(dreamList))  
 meanList.append(sum(dreamList)/len(dreamList))  
 tsList.append(df1.loc[i, 'ts'])  
 dreamList = []  
 disList.append(df1.loc[i, 'diseased'])  
 return meanList, minList, maxList, tsList, disList  
  
#возвращает время достижения min значения пульса за сон  
def relaxTime(df1):  
 dreamList = []  
 relaxList = []  
 tsList = []  
 disList = []  
 for i in range (len(df1) - 1):  
 dreamList.append(df1.loc[i, 'heart\_rate'])  
 tsList.append(df1.loc[i, 'ts'])  
 if (df1.loc[i+1, 'sleep\_number'] != df1.loc[i, 'sleep\_number']):  
 relaxIndex = dreamList.index(min(dreamList))  
 relaxValue = tsList[relaxIndex]  
 relaxSpeed = (tsList[relaxIndex] - tsList[0]).total\_seconds()/60/60  
 relaxList.append(relaxSpeed)  
 dreamList = []  
 tsList = []  
 disList.append(df1.loc[i, 'diseased'])  
 return relaxList, disList  
  
print(relaxTime(df1))  
#возвращает скорость между снами  
def speed(listHR, listTS):  
 df = []  
 checkList = []  
 for i in range (len(listHR) -1):  
 speedValue = (abs(listHR[i+1] - listHR[i])) / ((listTS[i+1]- listTS[i]).total\_seconds()/60/60)  
 df.append(speedValue)  
 if speedValue > 0.5:  
 checkList.append(listTS[i])  
 return df, checkList  
  
  
hrMeanDream1076, hrMinDream1076, hrMaxDream1076, tsDream1076, disDream1076 = checkDream(df1)  
hrMeanDream1062, hrMinDream1062, hrMaxDream1062, tsDream1062, disDream1062 = checkDream(df1062)  
hrMeanDream1073, hrMinDream1073, hrMaxDream1073, tsDream1073, disDream1073 = checkDream(df1073)  
print(tsDream1076)  
print(tsDream1062)  
print(tsDream1073)  
  
speedMean1076, listTS1076 = speed(hrMeanDream1076, tsDream1076)  
speedMin1076, listTS1076min = speed(hrMinDream1076, tsDream1076)  
speedMax1076, listTS1076max = speed(hrMaxDream1076, tsDream1076)  
  
speedMean1062, listTS1062 = speed(hrMeanDream1062, tsDream1062)  
speedMin1062, listTS1062min = speed(hrMinDream1062, tsDream1062)  
speedMax1062, listTS1062max = speed(hrMaxDream1062, tsDream1062)  
  
speedMean1073, listTS1073 = speed(hrMeanDream1073, tsDream1073)  
speedMin1073, listTS1073min = speed(hrMinDream1073, tsDream1073)  
speedMax1073, listTS1073max = speed(hrMaxDream1073, tsDream1073)  
  
# print('1076 скорость изменения среднего пульса за сон')  
# #print(hrMeanDream)  
# print(speedMean1076)  
print('1076 скорость изменения min пульса за сон')  
#print(hrMinDream)  
print(speedMin1076)  
# print('1076 скорость изменения max пульса за сон')  
# print(hrMaxDream)  
# print(speedMax)  
  
# print('1062 скорость изменения среднего пульса за сон')  
# #print(hrMeanDream1062)  
# print(speedMean1062)  
# print('1062 скорость изменения min пульса за сон')  
# # print(hrMinDream1062)  
# print(speedMin1062)  
# print('1062 скорость изменения max пульса за сон')  
# print(hrMaxDream1062)  
# print(speedMax1062)  
  
# print('1073 скорость изменения среднего пульса за сон')  
# #print(hrMeanDream1073)  
# print(speedMean1073)  
# print('1073 скорость изменения min пульса за сон')  
# # print(hrMinDream1073)  
# print(speedMin1073)  
# print('1073 скорость изменения max пульса за сон')  
# print(hrMaxDream1073)  
# print(speedMax1073)  
  
  
df\_roll = window(df\_scaled, week)  
#df1 = window(df1, week)  
  
  
#print('скользящие средние за неделю')  
#print(df\_roll.describe())  
#print('diseased')  
night = 391  
df\_min = window(df\_scaled, night)  
df\_min = window\_min(df\_min, 7)  
  
#возвращает список модулей разностей между элементами массива  
def difference(array):  
 df = []  
 for i in range (len(array)-1):  
 df.append(abs(array[i+1] - array[i]))  
 df.append(array[i+1] - array[i])  
 return df  
#возвращает дф разностей между элементами столбцов датафрейма  
def different(df\_scaled):  
 df\_dif = pd.DataFrame()  
 df\_dif['heart\_rate'] = difference(df\_scaled.heart\_rate)  
 df\_dif['hdr'] = difference(df\_scaled.hdr)  
 df\_dif['stress\_level'] = difference(df\_scaled.stress\_level)  
 df\_dif['sleep\_anxiety'] = difference(df\_scaled.sleep\_anxiety)  
 df\_dif['diseased'] = difference(df\_scaled.diseased)  
 return df\_dif  
  
#возвращает список дат когда отклонение было больше определенного  
def checkdate(dfx, val):  
 df = []  
 for i in range (len(dfx)):  
 if abs(dfx.heart\_rate[i]) > val:  
 df.append(dfx.ts[i])  
 return df  
  
#df\_dif1062 = different(df1026\_scaled)  
df\_dif = different(df\_scaled)  
df\_scaled["ts"] = df1.ts  
  
df\_scaled.set\_index('ts')  
df\_dif["ts"] = df1.ts  
df\_dif.set\_index('ts')  
###ПРОСТО ПРИНТ ДАТАФРЕЙМОВ  
# print('df\_dif')  
# print(df\_dif)  
# print('df\_scaled')  
# print(df\_scaled)  
  
df\_hr1076 = checkdate(df\_dif, 0.4)  
#df\_hr1026 = checkDream((df\_dif1062, 0.4))  
#dfmin\_hr = checkdate(df\_min, 0.6)  
  
#список дат когда отклонения было больше 0.4  
print('список дат когда отклонение пульса было больше 0.4')  
print(df\_hr1076)  
#  
  
#  
# print('Даты для 1076')  
# print(listTS1076)  
#  
# print('Даты для 1062')  
# print(listTS1062)  
#  
# print('ДАты для 1073')  
# print(listTS1073)  
  
#print(dfmin\_hr)  
#df\_scaled['MA'] = df\_scaled['heart\_rate'].rolling(window=week).mean()  
  
#plt.hist(df\_scaled.MA)  
x = np.linspace(-10, 10, 200)  
#plt.plot(df\_roll.hdr)  
  
  
#plt.boxplot(df1\_roll.heart\_rate)  
df\_roll.diseased = df\_roll.diseased  
df\_scaled.diseased = df\_scaled.diseased \* 0.5  
#df\_dif = df\_dif.set\_index('ts')  
  
  
# plt.figure(0)  
# #plt.gcf().autofmt\_xdate()  
# f, ax = plt.subplots(1)  
# #ax.set\_xticks(ax.get\_xticks()[::2])  
# df\_dif.plot(kind='line', x='ts', y='heart\_rate', color='red', ax=ax)  
# df\_scaled.plot(kind='line', x='ts', y='diseased', color='blue', ax=ax)  
#  
# #plt.show()  
#  
# # plt.plot(df1.ts, df\_dif.stress\_level)  
# # plt.plot(df\_scaled.ts, df\_scaled.diseased)  
# #plt.gcf().autofmt\_xdate()  
#  
# plt.plot(df\_dif.stress\_level)  
# plt.plot(df\_scaled.diseased)  
# # plt.plot(df\_dif.ts, df\_dif.stress\_level)  
# # plt.plot(df\_dif.ts, df\_scaled.diseased, color = 'orange')  
# plt.title("Распределение уровня стресса 1076 среднее за неделю")  
# plt.xlabel('дата наблюдения')  
# plt.ylabel('нормализованное значение')  
# plt.figure(1)  
#  
# #df\_dif.ts = df\_dif.ts.astype(str)  
# plt.plot(df\_dif.heart\_rate)  
# plt.plot(df\_scaled.diseased)  
# # plt.plot(df1.ts, df\_dif.heart\_rate)  
# # plt.plot(df1.ts, df\_scaled.diseased)  
# plt.title("Распределение уровня пульса 1076 min среди mean")  
# plt.ylabel('мин знач среди средних')  
# plt.xlabel('номер наблюдения')  
# #plt.figure(2)  
# #df\_dif = df\_dif.set\_index('ts')  
#  
# df.index = pd.to\_datetime(df.index)  
# df\_dif[['heart\_rate']].plot( figsize=(15,8))  
# df\_scaled['diseased'].plot(figsize = (15,8))  
  
#СКОРОСТИ MEAN  
# plt.figure(0)  
# plt.title('1076 скорость изменения среднего значения пульса за сон')  
# speedMean1076.append(mean(speedMean1076))  
# markers\_on = []  
# #пометка маркеров болезни на графиках  
# def markerDis(disL):  
# markers\_on = []  
# for i in range (len(disL)):  
# if disL[i] == 1:  
# markers\_on.append(i)  
# return markers\_on  
#  
# markers1076 = markerDis(disDream1076)  
# plt.plot(tsDream1076, speedMean1076, 'P-', label = 'mean', markevery = markers1076)  
# plt.title('1076 скорость изменения среднего значения пульса за сон')  
# plt.xlabel('дата сна')  
# plt.ylabel('скорость изменения среднего пульса (пульс в час)')  
# #plt.plot(disDream == '1', 'o--', label = 'dis')  
# #plt.plot(tsDream, speedMean)  
#  
#  
# plt.figure(1)  
# plt.title('1062 скорость изменения среднего значения пульса за сон')  
# speedMean1062.append(mean(speedMean1062))  
# markers1062 = markerDis(disDream1062)  
# plt.plot(tsDream1062, speedMean1062, 'P-', label = 'mean',markevery = markers1062)  
# plt.xlabel('дата сна')  
# plt.ylabel('скорость изменения среднего пульса (пульс в час)')  
#  
#  
# plt.figure(2)  
# plt.title('1073 скорость изменения среднего значения пульса за сон')  
# speedMean1073.append(mean(speedMean1073))  
# markers1073 = markerDis(disDream1073)  
# plt.plot(tsDream1073, speedMean1073, 'P-', label = 'mean', markevery = markers1073)  
# plt.xlabel('дата сна')  
# plt.ylabel('скорость изменения среднего пульса (пульс в час)')  
# print(markers1062)  
# print(markers1073)  
plt.figure(0)  
plt.title('1076 скорость изменения min значения пульса за сон')  
speedMax1076.append(mean(speedMax1076))  
markers\_on = []  
#пометка маркеров болезни на графиках  
def markerDis(disL):  
 markers\_on = []  
 for i in range (len(disL)):  
 if disL[i] == 1:  
 markers\_on.append(i)  
 return markers\_on  
  
markers1076 = markerDis(disDream1076)  
plt.plot(tsDream1076, speedMax1076, 'P-', label = 'mean', markevery = markers1076)  
#plt.plot(tsDream1076, speedMin1076)  
plt.title('1076 скорость изменения Max значения пульса за сон')  
plt.xlabel('дата сна')  
plt.ylabel('скорость изменения Max пульса (пульс в час)')  
#plt.plot(disDream == '1', 'o--', label = 'dis')  
#plt.plot(tsDream, speedMean)  
  
  
plt.figure(1)  
plt.title('1062 скорость изменения Max значения пульса за сон')  
speedMax1062.append(mean(speedMax1062))  
markers1062 = markerDis(disDream1062)  
plt.plot(tsDream1062, speedMax1062, 'P-', label = 'mean',markevery = markers1062)  
#plt.plot(speedMin1062)  
plt.xlabel('дата сна')  
plt.ylabel('скорость изменения Max пульса (пульс в час)')  
  
  
plt.figure(2)  
plt.title('1073 скорость изменения Max значения пульса за сон')  
speedMax1073.append(mean(speedMax1073))  
markers1073 = markerDis(disDream1073)  
plt.plot(tsDream1073, speedMax1073, 'P-', label = 'mean', markevery = markers1073)  
#plt.plot(speedMin1073)  
plt.xlabel('дата сна')  
plt.ylabel('скорость изменения Max пульса (пульс в час)')  
list1 = dreams(df1, 5)  
print(list1)  
print(list1.index(min(list1)))  
minmarker = list1.index(min(list1))  
print(minmarker)  
print(min(list1))  
plt.figure(3)  
plt.plot(list1, 'o-', label = 'min', markevery = minmarker)  
plt.title('Минимальное значение пульса за сон')  
plt.xlabel('номер значения')  
plt.ylabel('пульс')  
  
plt.figure(4)  
speedMark, disMark = relaxTime(df1)  
# speedMark = np.array(speedMark)  
# speedMark = speedMark.reshape(-1, 1)  
# speedMark = min\_max\_scaler.fit\_transform(speedMark)  
markerSpeed = markerDis(disMark)  
plt.title('Время релаксации во время снов 1062')  
plt.xlabel('номер значения')  
plt.ylabel('время, ч')  
plt.plot(tsDream1076, speedMark, 'o-', label = 'min', markevery = markerSpeed)  
plt.plot(tsDream1076, speedMark, 'o-', label = 'min', markevery = markerSpeed)  
  
plt.show()