### Ime, priimek:

LV05: Psihofiziološki signali in razpoznavanje stresa

# Uvod

Namen vaje:

* določiti značilke signalov, potrebne za strojno učenje
* izdelati model strojnega učenja, ki napoveduje nivo stresa na podlagi psihofizioloških signalov.

# Značilke podatkov in strojno učenje

Izračunali bomo značilke, ki jih bomo uporabljali pri strojnem učenju. Analizirali bomo njihove lastnosti.

Uporabljamo samo del podatkov, in sicer iz chest senzorja (na prsih)

1. ACC -> Accelerometer data

2. EDA -> Electrodermal Activity aka GSR (Galvanic Skin Response)

3. TEMP -> Skin Temperature (WESAD authors call it "body temperature")

## Nalaganje podatkov

DataManager.py omogoča uvoz in analizo podatkov.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import importlib

import os

import sys

module = os.path.abspath('D:\\P\_DEV\_PYTHON\\PROJECTS\\wesad\_experiments\\src\\main')

# module = os.path.abspath('/home/learner/wesad\_experiments/src/main')

# module = os.path.abspath("C:/Users\\18145\\development\\wesad\_experiments\\src\\main")

if module not in sys.path:

    sys.path.append(module)

from DataManager import DataManager

# Spremenljivka za podatkovni manager

# Ta nalozi samo podatke za chest senzor

manager = DataManager()

print(manager.BASELINE\_DATA)

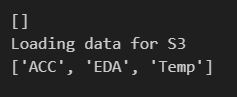
# Nalozimo podatke ene izbrane osebe 2..17

# metoda vrne 3 dictionaries podatkov za base, stress, amusement

# return base, stress, amusement

oseba = 3

data1 = manager.load(oseba) # load subject two data in two dicts, baseline and stress



# V data1 so trije podatki: za baseline stanje, stress in amusement

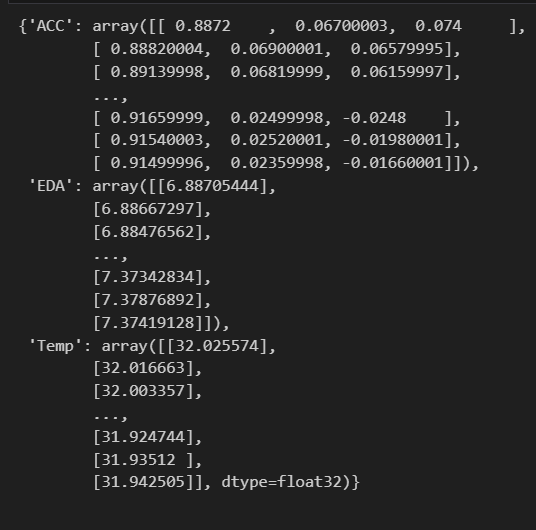
BASE = 0

STRESS = 1

AMUSEMENT = 2

# Poglejmo slovar za stres:

data1[STRESS]

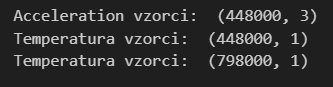


# Izpišimo število vzorcev senzorjev

print('Acceleration vzorci: ', data1[STRESS]['ACC'].shape)

print('Temperatura vzorci: ', data1[STRESS]['Temp'].shape)

print('Temperatura vzorci: ', data1[BASE]['Temp'].shape)



## Izračun značilk

Uporabljamo drseče okno, velikosti 42.000 vzorcev, kar ustreza času 60 sekund (vzorci so zajeti s 700 Hz).

Okno se pomika po 175 vzorcev (1/4 sekunde), torej se okna prekrivajo.

Splošne statistične značilke znotraj tega okna so:

1. Min and max value

2. Dinamično območje: Dynamic Range

3. Povprečje in standarda deviacija: Mean and STD

### Primer izračuna značilk

# Primer izračuna značilk za temperaturo (baseline)

temp = data1[BASE]['Temp']

window\_size = 42000

window\_shift = 175

max\_temp = []

min\_temp = []

dynamic\_range\_temp = []

for i in range(0,len(temp) - window\_size,window\_shift):

    window = temp[i:window\_size + i]

    max\_temp.append(np.amax(window))

    min\_temp.append(np.amin(window))

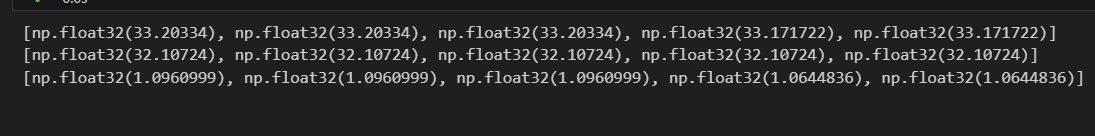
    dynamic\_range\_temp.append(max\_temp[-1] - min\_temp[-1])

# Izpišimo nekaj vrednosti

print(max\_temp[0:5])

print(min\_temp[0:5])

print(dynamic\_range\_temp[0:5])



## Izračun značilk izbranih oseb

Ponovno bomo naložili podatke za izbrane osebe, po njihovih številkah.

Izberemo številke oseb, in tipe signalov za nalaganje podatkov.

# Izberi subjekte

#manager.SUBJECTS = [2,3,4,5,6,7,8,9,10]

manager.SUBJECTS = [2,3]

print(manager.SUBJECTS)

# Izberi senzorje : ['ACC','ECG','EDA','EMG','Resp','Temp']

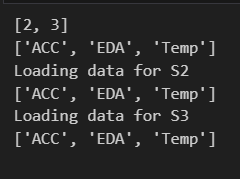
manager.RAW\_SENSOR\_VALUES = ['ACC', 'EDA', 'Temp']

print(manager.RAW\_SENSOR\_VALUES)

# Nalozi podatke vseh izbranih subjektov

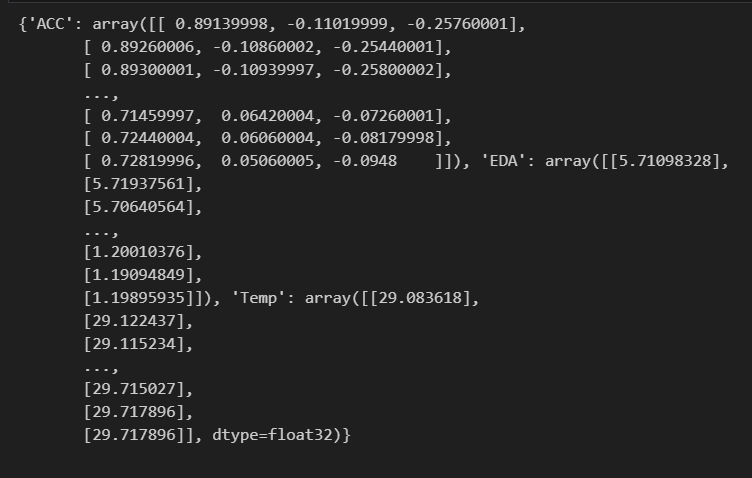
#print("Preparing data for model creation..")

manager.load\_all(subjects=manager.SUBJECTS)



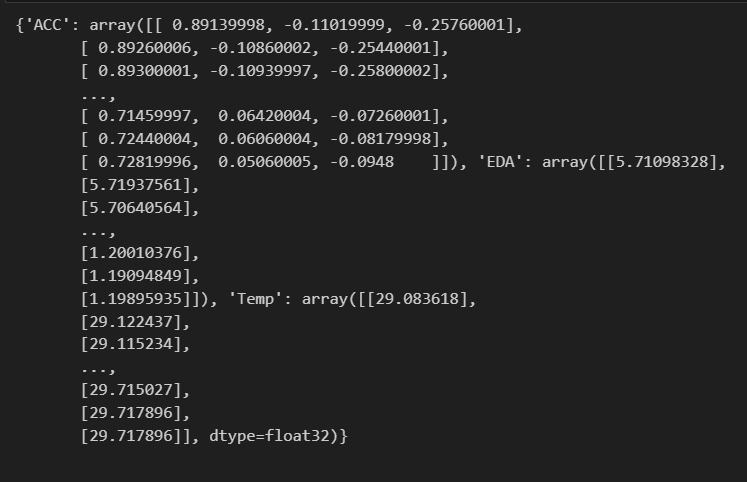
# Preverimo, za koliko oseb imamo podatke:

print("Oseb: ", len(DataManager.BASELINE\_DATA))



# To je seznam, več slovarjev za vsako osebo, za BASELINE stanje

print(DataManager.BASELINE\_DATA[0])



### Izračun značilk za naložene podatke

from Features2 import decompose\_eda, make\_target, ACC\_features, MSRS, Detect\_peaks\_ECG, SCRL, EMG

BASE = 0

STRESS = 1

AMUSEMENT = 2

numb\_of\_measures\_4\_HZ = 21864   #data\_new[b'signal'][b'wrist'][b'EDA'].shape[0]

window\_size\_ts = 5

number\_o\_in\_0\_25\_sec = 175

window\_size\_o =  number\_o\_in\_0\_25\_sec \* 4 \* window\_size\_ts

timestep\_re = numb\_of\_measures\_4\_HZ \* 0.25/3826200.0

manager.RAW\_SENSOR\_VALUES = ['ACC', 'Temp' ] #, 'EDA', 'Temp']

data = []

TRAIN\_SUBJECTS = [2,3]

person\_ind = 0

# Preko vseh oseb

for subject in TRAIN\_SUBJECTS:

    person\_ind = subject - 2

    print('++++++++++++++++++++++      Znacilke oseba stev. ', person\_ind, ' SUBJECT ', subject  )

#for person in [0]:

    person\_feat = pd.DataFrame()

    #for sensor in manager.RAW\_SENSOR\_VALUES:

# ACC\_Chest

    # ACC = data\_new[b'signal'][b'chest'][b'ACC']

    sensor = 'ACC'

    data\_stres = manager.STRESS\_DATA[person\_ind][sensor]

    num\_samples = data\_stres.shape[0]

    print('STRES:    Person:', person\_ind, 'Sensor: ', sensor, ' Num\_Samples: ', data\_stres.shape)

    data\_base = manager.BASELINE\_DATA[person\_ind][sensor]

    num\_samples = data\_base.shape[0]

    print('BASELINE: Person:', person\_ind, 'Sensor: ', sensor, ' Num\_Samples: ', data\_base.shape)

    ACC\_stres = pd.DataFrame(ACC\_features( data\_stres, window\_size\_o,number\_o\_in\_0\_25\_sec,timestep\_re,'\_chest'))

    ACC\_base = pd.DataFrame(ACC\_features( data\_base, window\_size\_o,number\_o\_in\_0\_25\_sec,timestep\_re,'\_chest'))

    ACC\_feat = pd.concat([ACC\_base, ACC\_stres], axis=0)

    print('ACC\_stres : ', ACC\_stres.shape)

    print('ACC\_base  : ', ACC\_base.shape)

# if 1 :

    sensor = 'EDA'

    data\_s = manager.STRESS\_DATA[person\_ind][sensor]

    data\_b = manager.BASELINE\_DATA[person\_ind][sensor]

    #EDA\_chest

    EDA\_stres = pd.DataFrame(MSRS(data\_s,'EDA\_chest',700))

    EDA\_base = pd.DataFrame(MSRS(data\_b,'EDA\_chest',700))

# Temperatura

    #if sensor == 'Temp':

    sensor = 'Temp'

    data\_stres = manager.STRESS\_DATA[person\_ind][sensor]

    data\_base = manager.BASELINE\_DATA[person\_ind][sensor]

    print('STRES:    Person:', person\_ind, 'Sensor: ', sensor, ' Num\_Samples: ', data\_stres.shape)

    print('BASELINE: Person:', person\_ind, 'Sensor: ', sensor, ' Num\_Samples: ', data\_base.shape)

    Temp\_stres = pd.DataFrame(MSRS(data\_stres,'Temp\_chest', 700))

    Temp\_base = pd.DataFrame(MSRS(data\_base,'Temp\_chest', 700))

    print('Temp\_stres : ', Temp\_stres.shape)

    print('Temp\_base  : ', Temp\_base.shape)

    Temp\_feat = pd.concat([Temp\_base, Temp\_stres], axis=0)

    #person\_base = pd.concat([person\_baseline, Temp\_base], axis=1)

    # ZNACILKE ENE OSEBE : zdruzi po stolpcih

    person\_feat\_base = pd.concat([ACC\_base, Temp\_base, EDA\_base], axis=1)

    person\_feat\_base['Label'] = BASE

    person\_feat\_base['Person'] = subject

    person\_feat\_stres = pd.concat([ACC\_stres, Temp\_stres, EDA\_stres], axis=1)

    person\_feat\_stres['Label'] = STRESS

    person\_feat\_stres['Person'] = subject

    person\_feat = pd.concat([person\_feat\_base, person\_feat\_stres], axis=0)

    print('-------- person\_feat : ', person\_feat.shape)

    # DROP NAN

    #nanloc = person\_feat['mean\_Temp\_chest'].notna()

    person\_feat.dropna(axis = 0, inplace=True)

    #person\_feat.info()

    if person\_ind==0:

        data = person\_feat

    else:

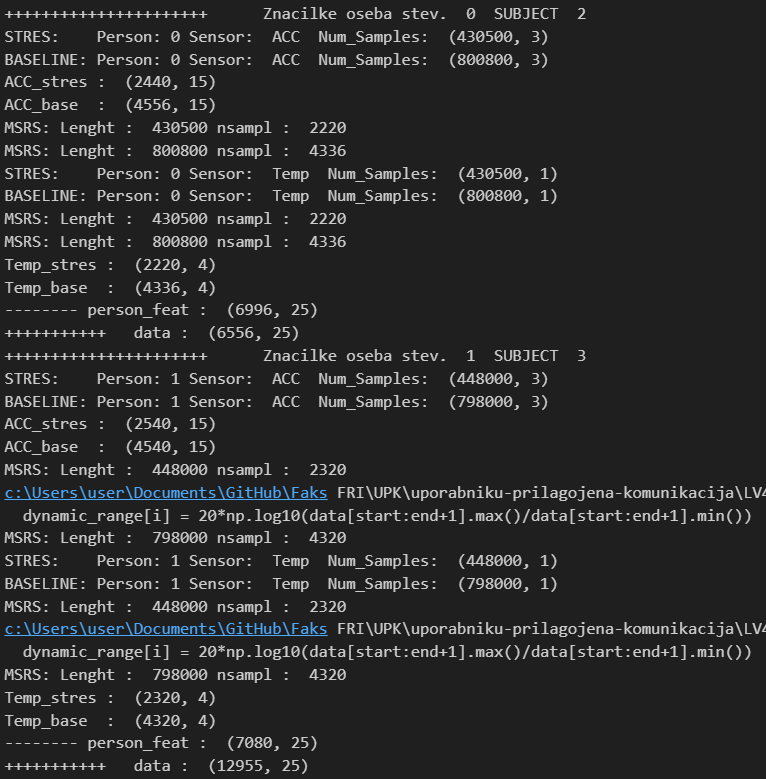
        data = pd.concat([data, person\_feat])

    #print(ACC\_data\_chest.head())

    print('+++++++++++   data : ', data.shape)

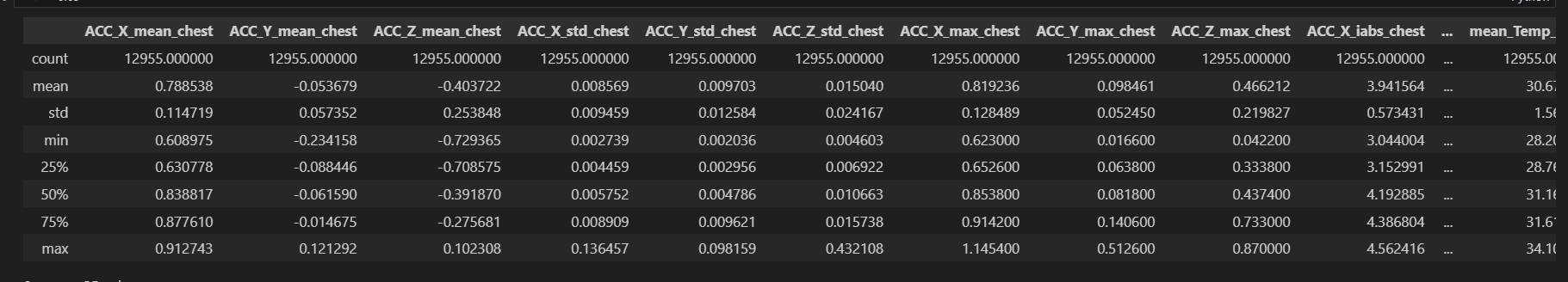
    person\_ind = person\_ind + 1

data.info()



### Preglejte izdelane značilke

data.describe()



## Vizualizacija značilk

Koliko se razlikujejo features podatki za obe stanji, med osebami ?

### Izris kde grafa po osebah za stanji 0, 1, za izbrano značilko

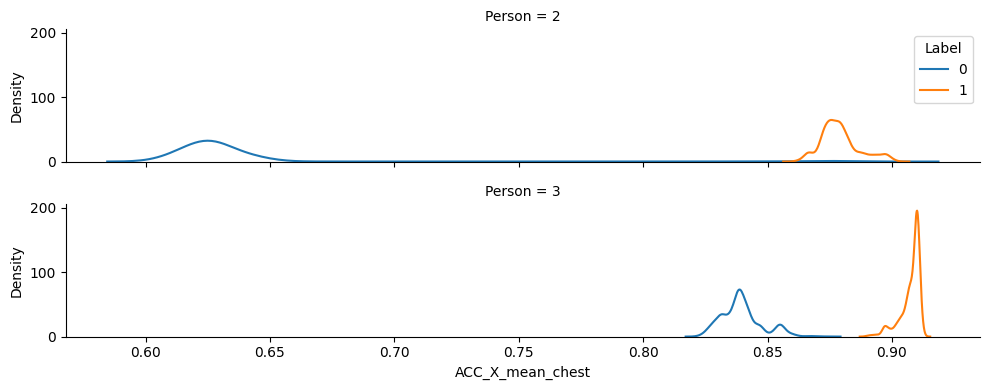
import seaborn as sns

import matplotlib.pyplot as plt

fg1 = sns.FacetGrid(data, row='Person', hue="Label", aspect=5, legend\_out=False, height=2)

fg1.map(sns.kdeplot,'ACC\_X\_mean\_chest').add\_legend()

plt.show()



Izriši še za kakšno dodatno značilko. Ali se histogram značilke razlikuje glede na stanje osebe, ali lahko torej iz te značilke ločimo stanji ?

### Izris skupne porazdelitve po osebah

import seaborn as sns

import matplotlib.pyplot as plt

print(data.shape)

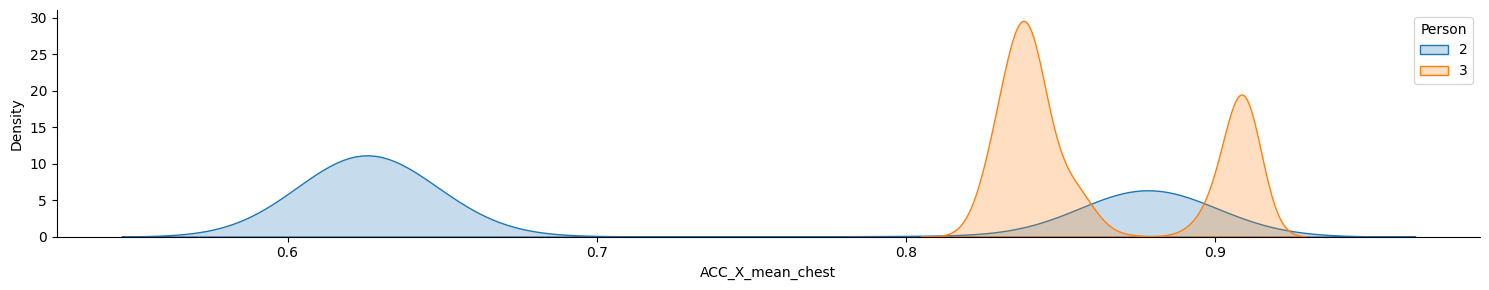
for col in data.columns[0:10]:

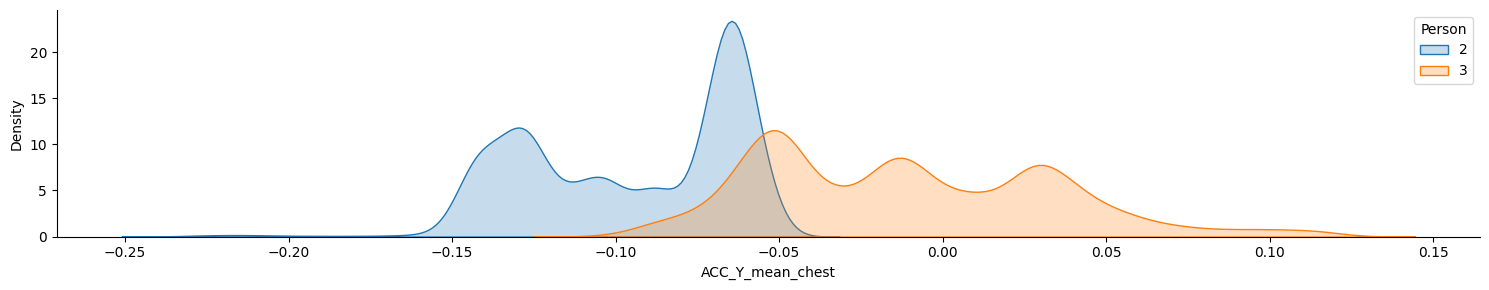
    fg = sns.FacetGrid(data, hue="Person", aspect=5, legend\_out=False)

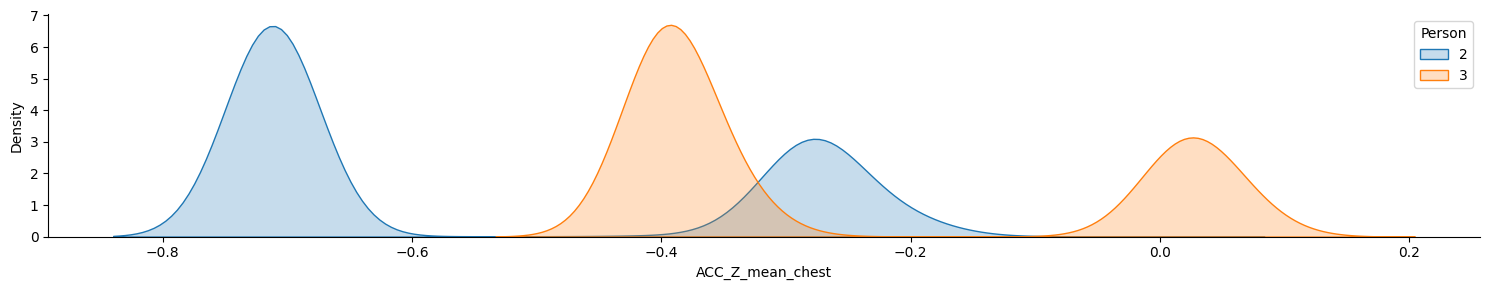
    fg.map(sns.kdeplot, col, fill=True).add\_legend()

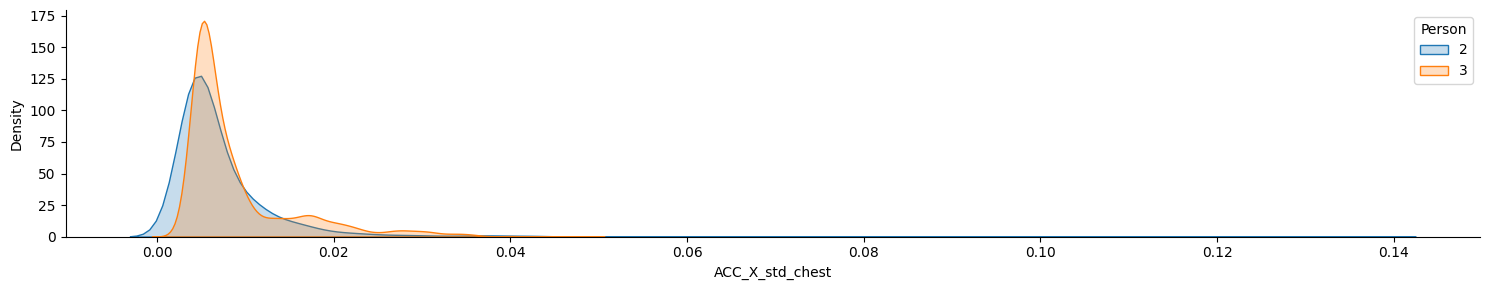
plt.show()

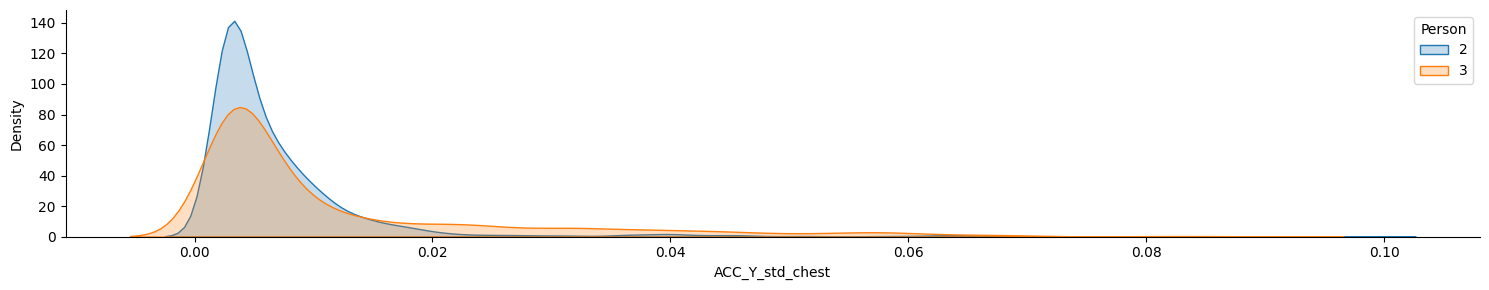
Komentiraj grafe.

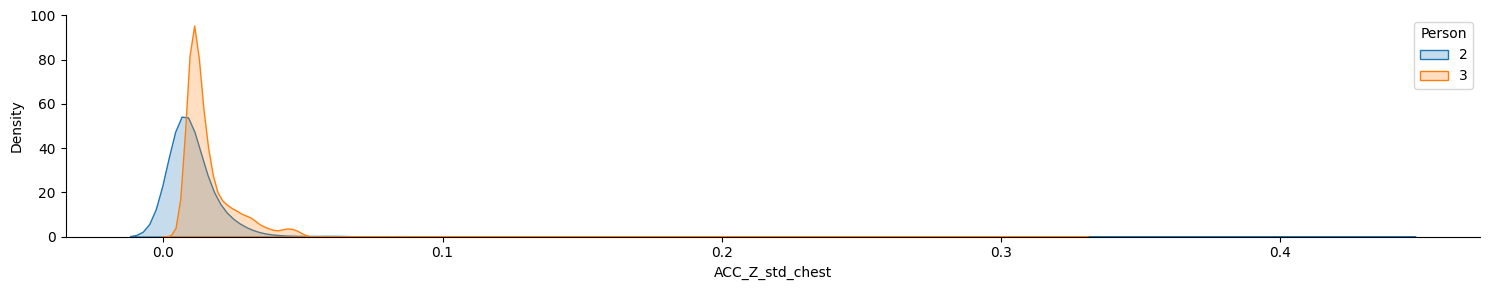


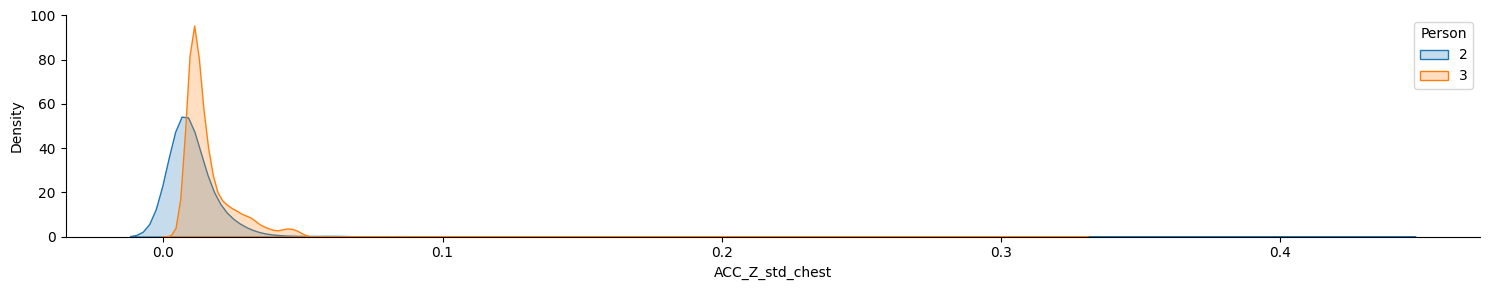


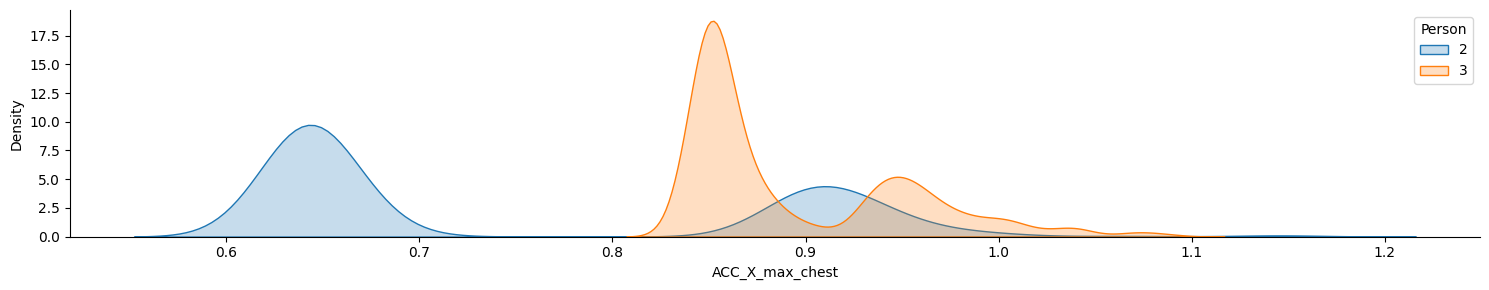


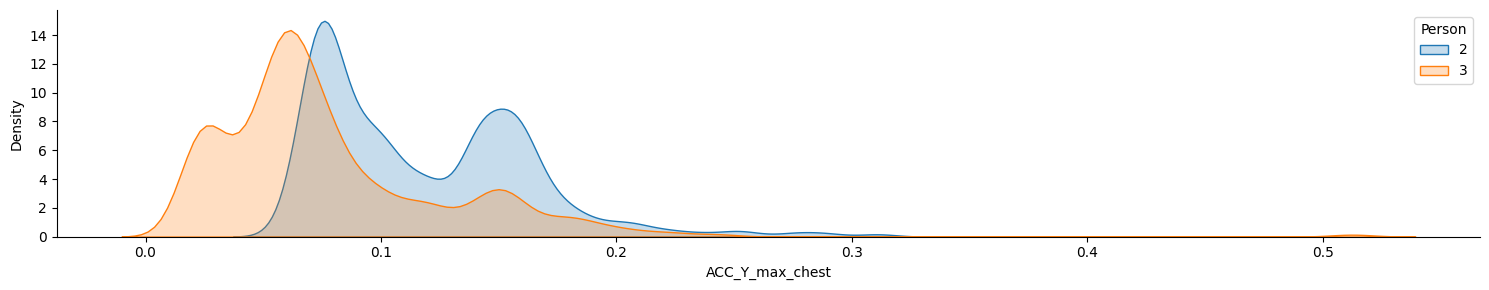


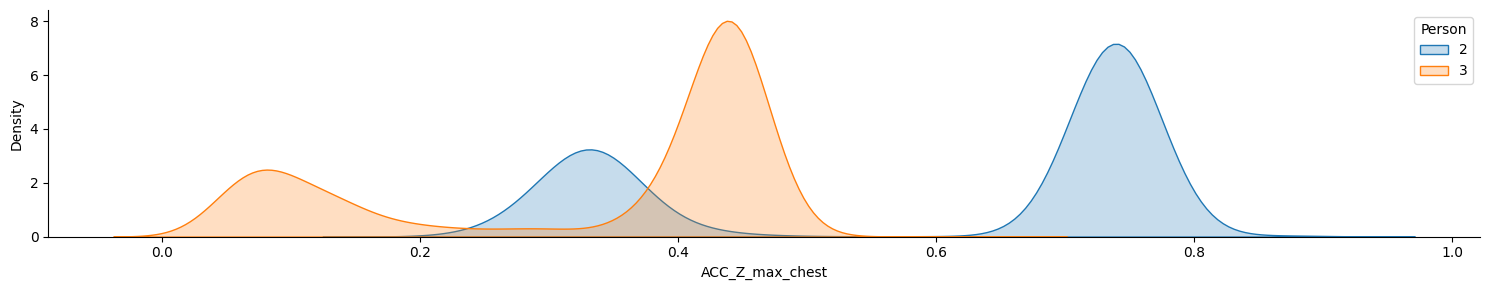


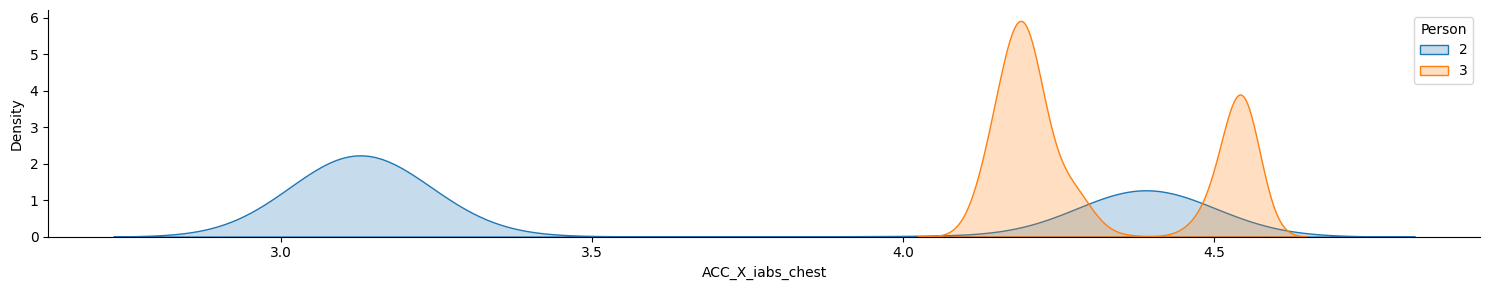












### Izris skupne porazdelitve vseh oseb, po stanjih

Katere značilke se porazdelitve najbolj razlikujejo za obe stanji ?

import seaborn as sns

import matplotlib.pyplot as plt

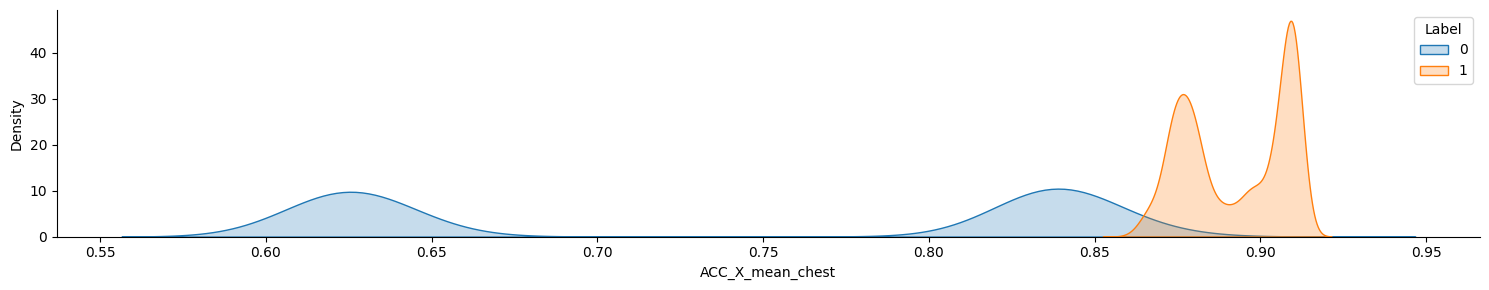
print(data.shape)

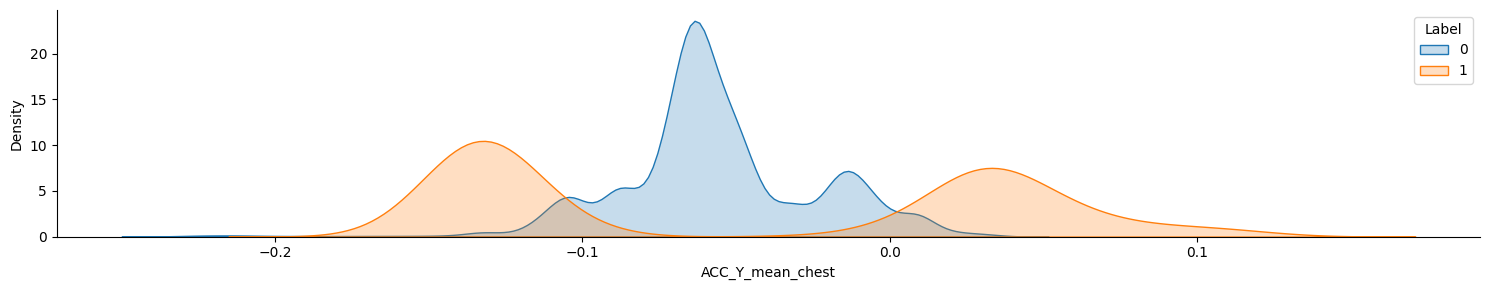
for col in data.columns[0:10]:

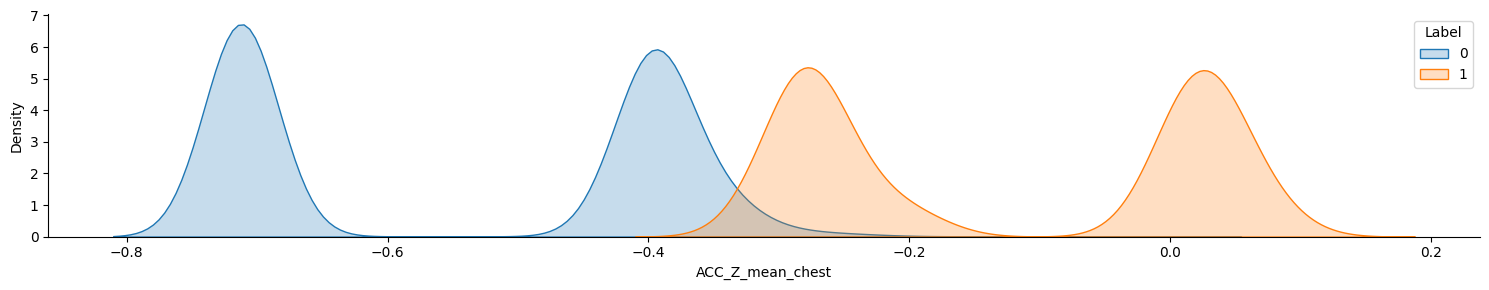
    fg = sns.FacetGrid(data, hue="Label", aspect=5, legend\_out=False)

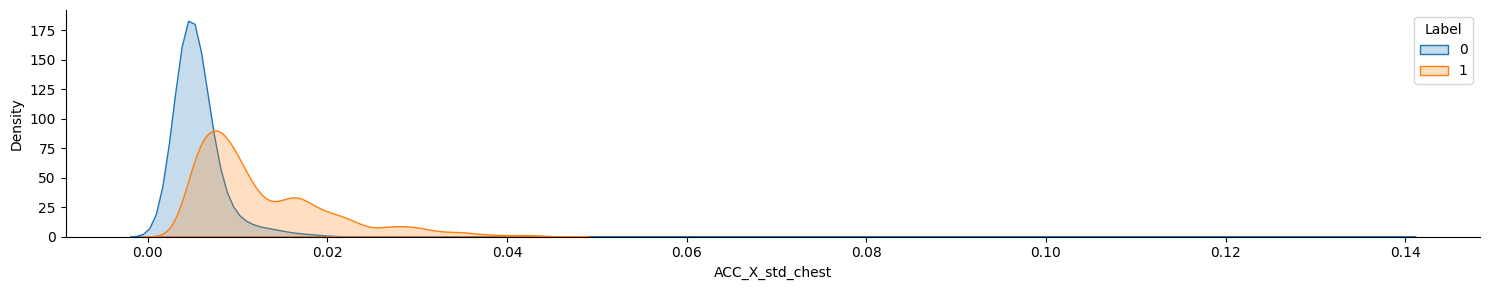
    fg.map(sns.kdeplot, col, fill=True).add\_legend()

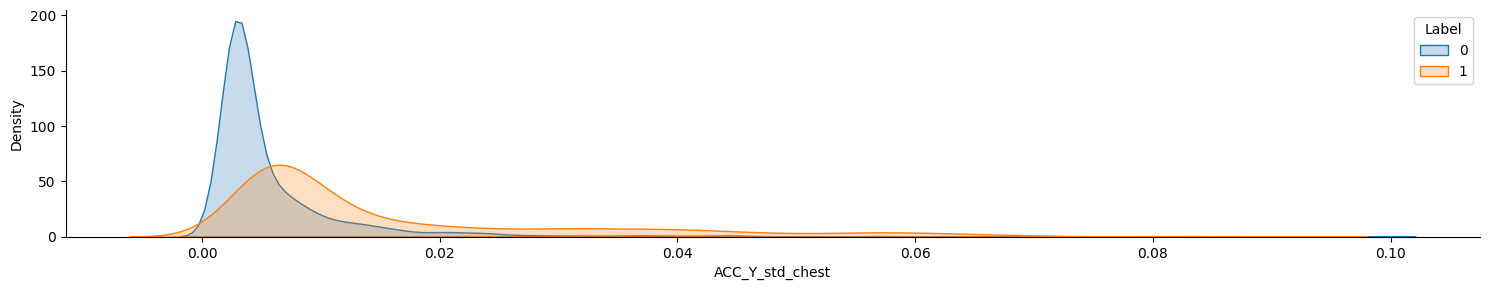
plt.show()

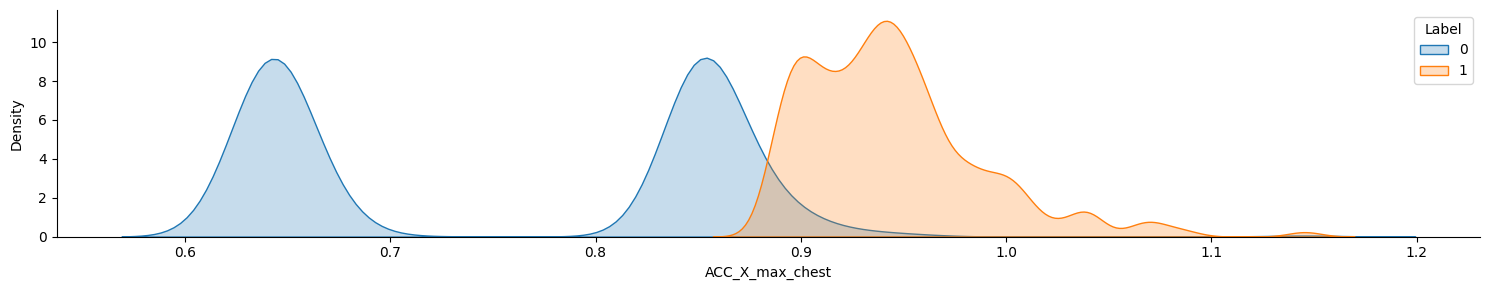


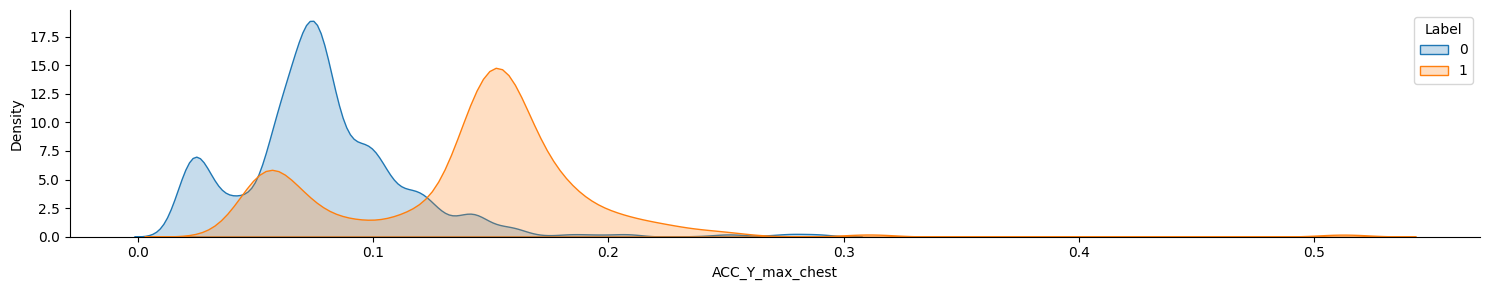


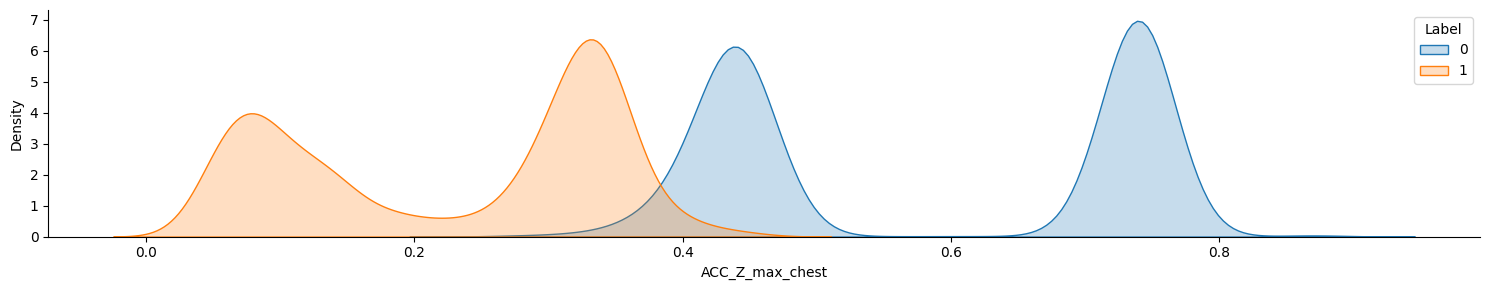


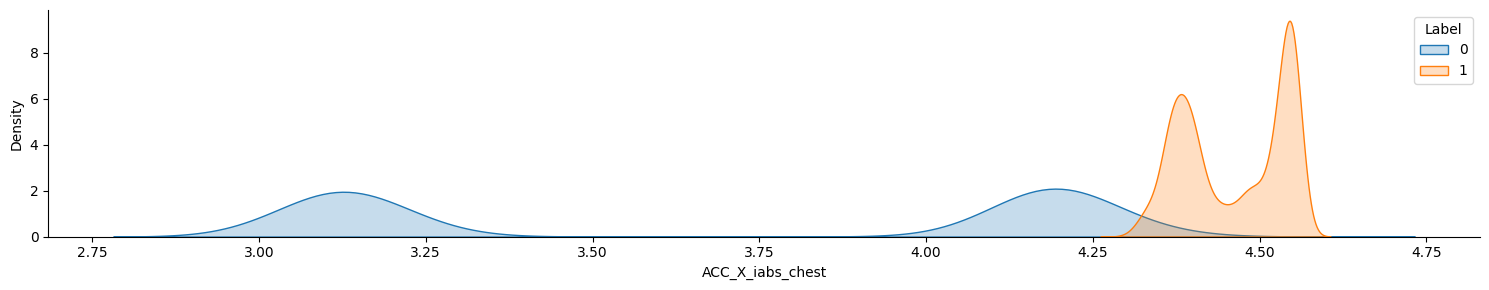












## Normalizirani podatki

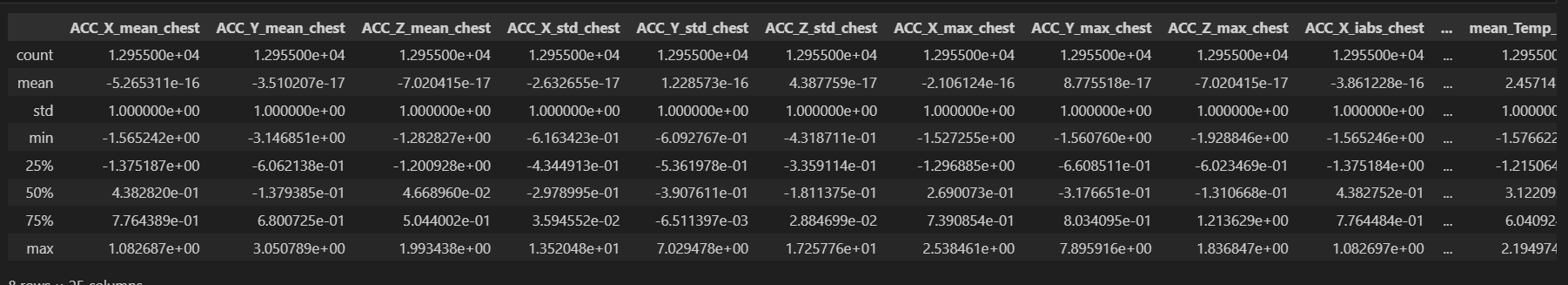
### Normalizacija podatkov

norm\_data=(data - data.mean())/data.std()

norm\_data['Label'] = data['Label']

norm\_data['Person'] = data['Person']

norm\_data.describe()



Podatki se ne bodo bistven razlikovali ker smo jih normalizirali

Izris porazdelitev:

Grafi izgledjo enako ker nismo še med sabo oseb porazdelili.

for col in norm\_data.columns[0:19]:

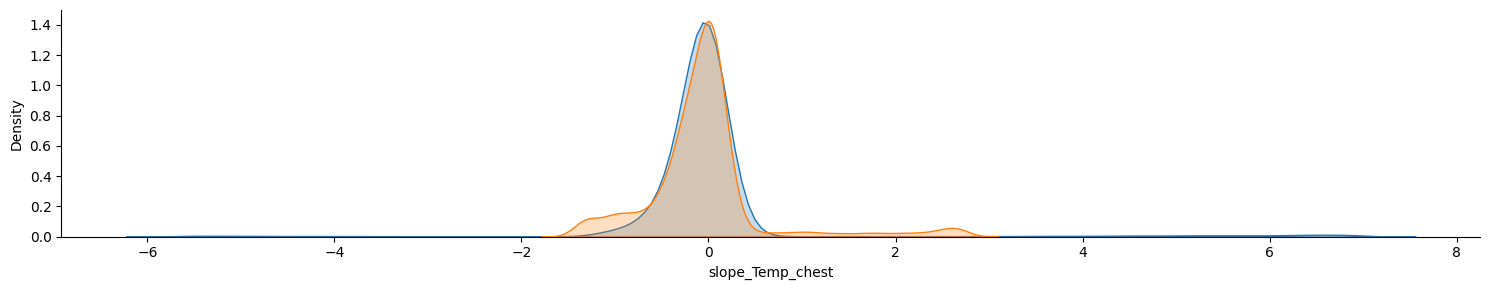
    fg2 = sns.FacetGrid(norm\_data, hue="Label", aspect=5)

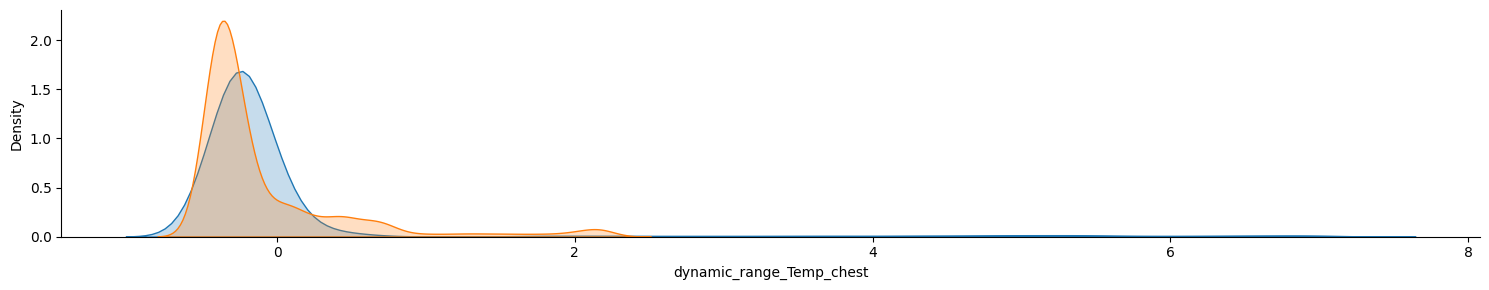
    fg2.map(sns.kdeplot, col, fill=True)

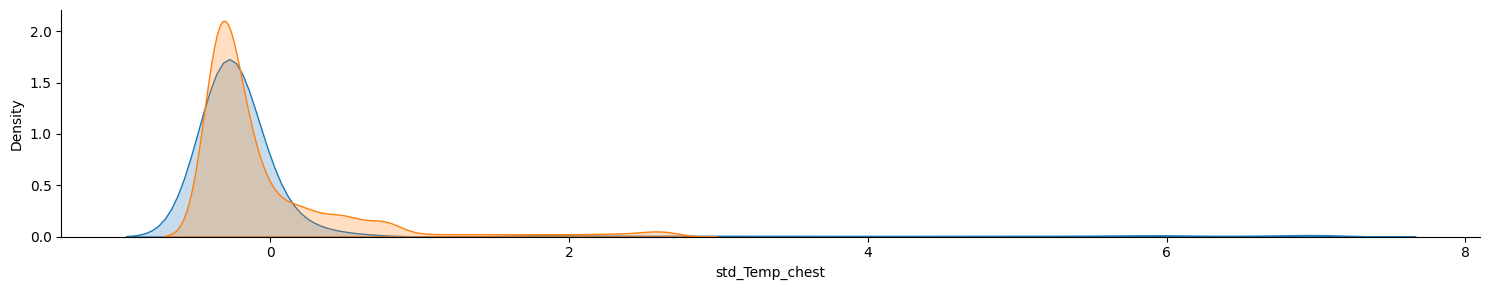
plt.show()

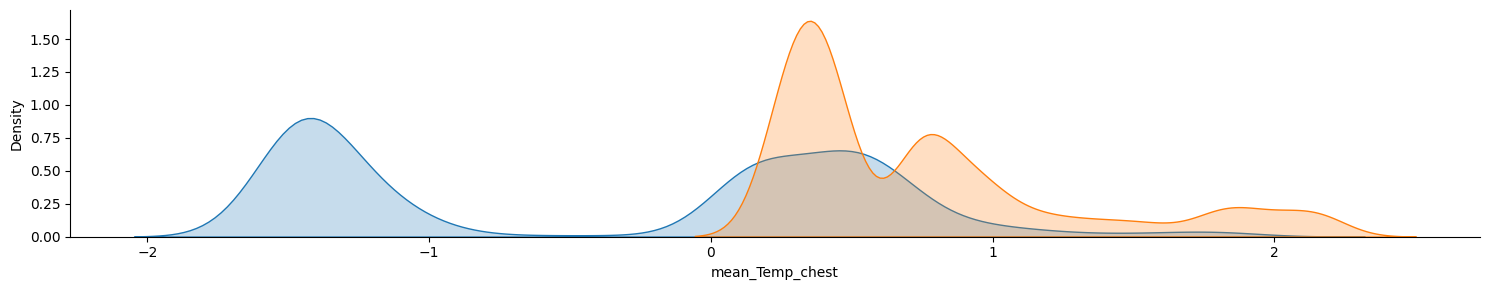
**Primerjal sem bazo (modra) in stres (oranžna) stanja. Bolj kot se podatkovne točke prekrivajo, manj primerna je značilka za razlikovanje med tema dvema stanji. Če so podatkovne točke za bazo in stresna stanja tesno skupaj, to pomeni, da značilka ni primerna za zaznavanje razlik med tema stanjema. To nakazuje, da je bila oseba pod stresom, vendar ta značilka ni bila učinkovita pri zaznavanju značilnosti stresa.**

**Primerjamo bazo in stresna stanja, da ocenimo učinkovitost značilk pri razlikovanju med tema dvema pogojema.**









# Strojno učenje modela prepoznave stanja (stres, bazično)

## Razdelitev podatkov na učno in testno množico (naključno)

Naključna razdelitev podatkov

from sklearn.model\_selection import train\_test\_split

# TESTNI IN UCNI SET PODATKOV

# Uporaba vseh oseb Person v train in test set.

# Oznaka razreda, labela stanja

norm\_y = norm\_data['Label']

data\_y = data['Label']

# Features, podatki

norm\_x = norm\_data.drop(['Label','Person'],axis=1)

data\_x = data.drop(['Label','Person'],axis=1)

norm\_x.shape

# Uporaba originalnih podatkov

#x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x, data\_y, test\_size=0.3)

# Uporaba normaliziranih podatkov

x\_train,x\_test,y\_train,y\_test = train\_test\_split(norm\_x, norm\_y, test\_size=0.3)

Naloga: Ugotovite, koliko vzorcev imamo v učni ter koliko v testni množici. Uporabite describe metodo

Train: 13738 rows x 23 columns Test: 5889 rows x 23 columns.

## Učenje modela in testiranje točnosti

**Na testni množici izračunamo pravilnost napovedi modela, vstavite in komentirajte rezultate.**

import sklearn as sklearn

from sklearn.neighbors import KNeighborsClassifier

import sklearn.linear\_model as skl\_lm

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.discriminant\_analysis import QuadraticDiscriminantAnalysis

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.metrics import roc\_curve

from sklearn.metrics import classification\_report

print(' y\_train  : ', y\_train.shape)

print(' x\_train  : ', x\_train.shape)

print(' x\_test  : ', x\_test.shape)

clf = skl\_lm.LogisticRegression(solver='newton-cg', penalty='l2', max\_iter=1000)

clf.fit(x\_train,y\_train)

y\_out = clf.predict(x\_test)

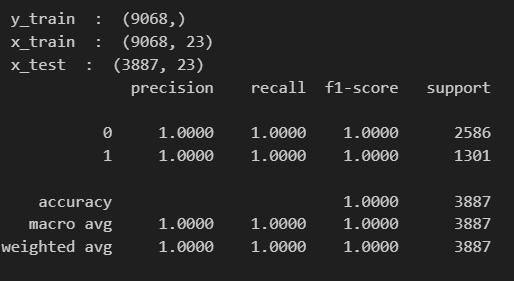
print(classification\_report(y\_test, y\_out, digits=4))

# Confusion matrix

disp = ConfusionMatrixDisplay(confusion\_matrix(y\_test, y\_out, normalize=None))

disp.plot()

Rezultat in komentar pravilnosti razpoznave:



Skupna natančnost:

Accuracy: 1.0000

Povprečje (macro avg): 1.0000

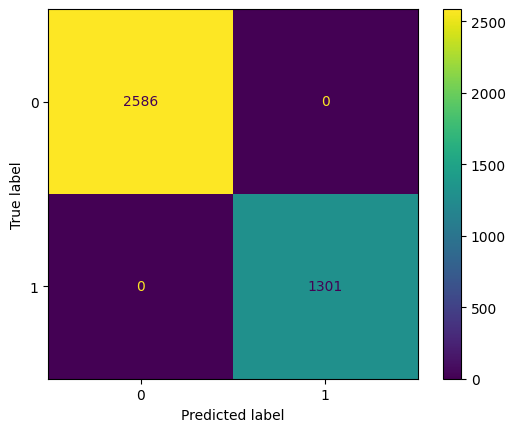
Tehtano povprečje (weighted avg): 1.0000

Skupaj testiranih primerov: 3887

Matrika zmede

Matrika zmede nam pomaga razumeti, kako dobro model klasificira posamezne razrede. V tem primeru so vse vrednosti pravilno klasificirane, kar nakazuje zelo dobro delovanje modela.

Opomba o optimizaciji in natančnosti:

Ker se podatki za treniranje in testiranje prekrivajo (isti primeri so prisotni v obeh naborih), rezultat lahko deluje pretirano optimistično. Takšna postavitev lahko vodi v "pretirano prileganje" (overfitting), kjer model preveč natančno napove primere, ki jih je že videl.

## Dodatno: Učna množica izbran osebe, test na drugi osebi

Sedaj učimo na nekaj izbranih osebah.

Model testiramo na novi osebi.

Ugotovite, kakšni so rezultati (točnost napovedi).

from sklearn.model\_selection import train\_test\_split

# TESTNI IN UCNI SET PODATKOV

# Uporaba Izbranih Person v train in drugih v test setu

TRAIN\_PERSONS = [2]

TEST\_PERSON = 3

izbira\_train = data['Person']==0

# Preko vseh TRAIN oseb

for subject in TRAIN\_PERSONS:

    izbira = data['Person']==subject

    izbira\_train = izbira\_train | izbira

print(izbira\_train)

norm\_train = data[izbira\_train]

norm\_train.info()

#

# Izbira testne osebe

izbira\_test = data['Person']==TEST\_PERSON

norm\_test = data[izbira\_test]

norm\_test.info()

# Naredi train set

y\_train\_data = norm\_train['Label']

x\_train\_data = norm\_train.drop(['Label','Person'],axis=1)

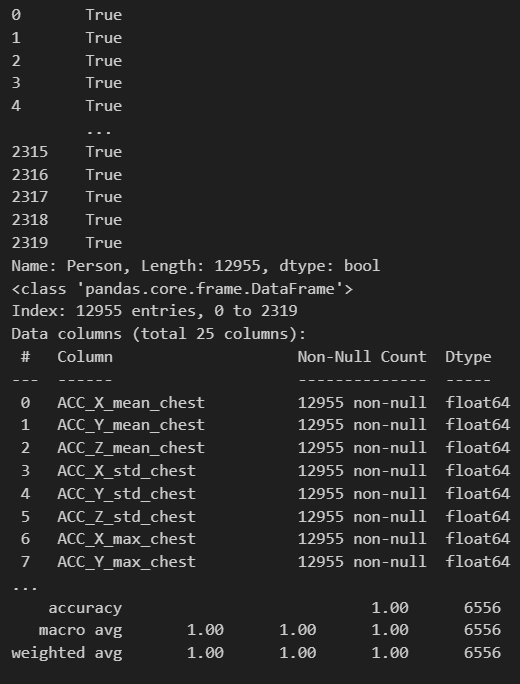
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_train\_data, y\_train\_data, test\_size=0.8)

# Test set

y\_test = norm\_test['Label']

x\_test = norm\_test.drop(['Label','Person'],axis=1)

Nato sledi učenje in testiranje. Vstavite rezultate in komentirajte pravilnost razpoznave.



- Vsi podatki za treniranje so označeni z `True`, kar pomeni, da so bili vsi podatki izbrani za treniranje.

- Skupno število vnosov za treniranje je 12,955.

- Podatki vsebujejo 12,955 vnosov in 25 stolpcev.

- Vsi stolpci imajo ne-null vrednosti.

- Stolpci vključujejo različne meritve, kot so `ACC\_X\_mean\_chest`, `mean\_Temp\_chest`, `mean\_EDA\_chest`, itd.

- Stolpec `Label` je ciljna spremenljivka, stolpec `Person` pa označuje osebo.

- Podatki vsebujejo 6,556 vnosov in 25 stolpcev.

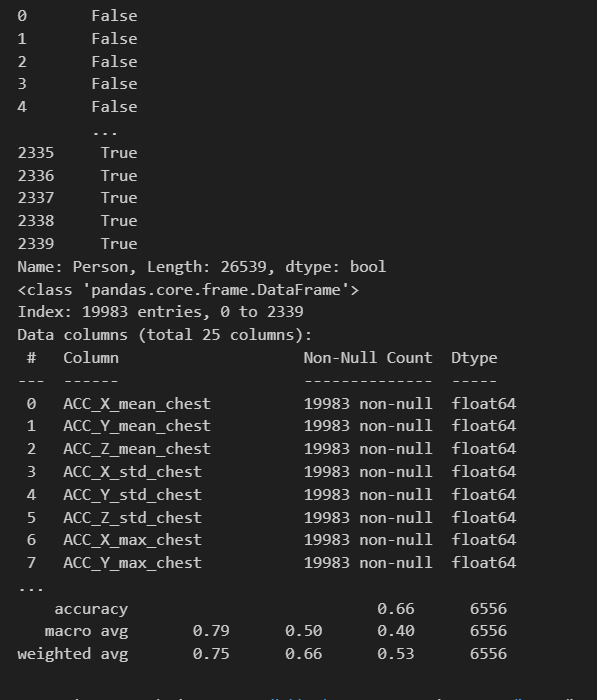
- Vsi stolpci imajo ne-null vrednosti.

- Stolpci so enaki kot pri podatkih za treniranje.

- Podrobno poročilo o klasifikaciji kaže, da je model dosegel popolno natančnost (`precision`), priklic (`recall`) in F1-oceno (`f1-score`) za obe razredi (0 in 1).

- Skupna natančnost (`accuracy`) je 100%.

To kaže, da je model zelo uspešen pri razpoznavi podatkov, vendar je pomembno preveriti, ali so podatki uravnoteženi in ali model ni preveč prilagojen (**OVERFITTING!!!**).



Ko ne vključimo testa v train, dobimo bolj relevantne rezultate.

Iz teh podatkov lahko razberemo naslednje:

- Pravilnost razpoznave je 66.29%, kar pomeni, da model ni zelo natančen.

- Podrobno poročilo o klasifikaciji kaže, da je model dosegel natančnost (`precision`) 66% za razred 0 in 92% za razred 1, vendar je priklic (`recall`) za razred 1 zelo nizek (0%).

- Skupna natančnost (`accuracy`) je 66%.

To kaže, da model ni zelo uspešen pri razpoznavi podatkov, še posebej pri razredu 1. Potrebno bi bilo izboljšati predobdelavo podatkov, izbrati boljši model ali prilagoditi parametre modela za boljše rezultate.