Ime, priimek:

LV05: Psihofiziološki signali in razpoznavanje stresa

1 Uvod

Namen vaje:

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

2 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")

2.1 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
```

```
[]
Loading data for S3
['ACC', 'EDA', 'Temp']
```

```
# V data1 so trije podatki: za baseline stanje, stress in amusement
BASE = 0
STRESS = 1
AMUSEMENT = 2

# Poglejmo slovar za stres:
data1[STRESS]
```

```
{'ACC': array([[ 0.8872    ,  0.06700003,  0.074     ],
       [ 0.88820004,  0.06900001,  0.06579995],
       [ 0.89139998,  0.06819999,  0.06159997],
       ...,
       [ 0.91659999,  0.02499998, -0.0248    ],
       [ 0.91540003,  0.02520001, -0.01980001],
       [ 0.91499996,  0.02359998, -0.01660001]]),
 'EDA': array([[6.88705444],
       [6.88667297],
       [6.88476562],
       ...,
       [7.37342834],
       [7.37876892],
       [7.37419128]]),
 'Temp': array([[32.025574],
       [32.016663],
       [32.003357],
       ...,
       [31.924744],
       [31.93512 ],
       [31.942505]], dtype=float32)}
```

```
# 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)
```

```
Acceleration vzorci:  (448000, 3)
Temperatura vzorci:  (448000, 1)
Temperatura vzorci:  (798000, 1)
```

2.2 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

2.2.1 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])
```

```
[np.float32(33.20334), np.float32(33.20334), np.float32(33.20334), np.float32(33.171722), np.float32(3
[np.float32(32.10724), np.float32(32.10724), np.float32(32.10724), np.float32(32.10724), np.float32(32
[np.float32(1.0960999), np.float32(1.0960999), np.float32(1.0960999), np.float32(1.0644836), np.float3
```

2.3 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)
```

```
[2, 3]
['ACC', 'EDA', 'Temp']
Loading data for S2
['ACC', 'EDA', 'Temp']
Loading data for S3
['ACC', 'EDA', 'Temp']
```

```
# Preverimo, za koliko oseb imamo podatke:
print("Oseb: ", len(DataManager.BASELINE_DATA))
```

```
{'ACC': array([[ 0.89139998, -0.11019999, -0.25760001],
       [ 0.89260006, -0.10860002, -0.25440001],
       [ 0.89300001, -0.10939997, -0.25800002],
       ...,
       [ 0.71459997,  0.06420004, -0.07260001],
       [ 0.72440004,  0.06060004, -0.08179998],
       [ 0.72819996,  0.05060005, -0.0948    ]]), 'EDA': array([[5.71098328]
       [5.71937561],
       [5.70640564],
       ...,
       [1.20010376],
       [1.19094849],
       [1.19895935]]), 'Temp': array([[29.083618],
       [29.122437],
       [29.115234],
       ...,
       [29.715027],
       [29.717896],
       [29.717896]], dtype=float32)}
```

```
# To je seznam, več slovarjev za vsako osebo, za BASELINE stanje
print(DataManager.BASELINE_DATA[0])
```

```
{'ACC': array([[ 0.89139998, -0.11019999, -0.25760001],
       [ 0.89260006, -0.10860002, -0.25440001],
       [ 0.89300001, -0.10939997, -0.25800002],
       ...,
       [ 0.71459997,  0.06420004, -0.07260001],
       [ 0.72440004,  0.06060004, -0.08179998],
       [ 0.72819996,  0.05060005, -0.0948    ]]), 'EDA': array([[5.71098328],
       [5.71937561],
       [5.70640564],
       ...,
       [1.20010376],
       [1.19094849],
       [1.19895935]]), 'Temp': array([[29.083618],
       [29.122437],
       [29.115234],
       ...,
       [29.715027],
       [29.717896],
       [29.717896]], dtype=float32)}
```

2.3.1 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()
```

```
++++++++++++++++++++++      Znacilke oseba stev.  0  SUBJECT  2
STRES:    Person: 0 Sensor:  ACC  Num_Samples:  (430500, 3)
BASELINE: Person: 0 Sensor:  ACC  Num_Samples:  (800800, 3)
ACC_stres :  (2440, 15)
ACC_base  :  (4556, 15)
MSRS: Lenght :  430500 nsampl :  2220
MSRS: Lenght :  800800 nsampl :  4336
STRES:    Person: 0 Sensor:  Temp  Num_Samples:  (430500, 1)
BASELINE: Person: 0 Sensor:  Temp  Num_Samples:  (800800, 1)
MSRS: Lenght :  430500 nsampl :  2220
MSRS: Lenght :  800800 nsampl :  4336
Temp_stres :  (2220, 4)
Temp_base  :  (4336, 4)
-------- person_feat :  (6996, 25)
++++++++++   data :  (6556, 25)
++++++++++++++++++++++      Znacilke oseba stev.  1  SUBJECT  3
STRES:    Person: 1 Sensor:  ACC  Num_Samples:  (448000, 3)
BASELINE: Person: 1 Sensor:  ACC  Num_Samples:  (798000, 3)
ACC_stres :  (2540, 15)
ACC_base  :  (4540, 15)
MSRS: Lenght :  448000 nsampl :  2320
c:\Users\user\Documents\GitHub\Faks FRI\UPK\uporabniku-prilagojena-komunikacij
  dynamic_range[i] = 20*np.log10(data[start:end+1].max()/data[start:end+1].min
MSRS: Lenght :  798000 nsampl :  4320
STRES:    Person: 1 Sensor:  Temp  Num_Samples:  (448000, 1)
BASELINE: Person: 1 Sensor:  Temp  Num_Samples:  (798000, 1)
MSRS: Lenght :  448000 nsampl :  2320
c:\Users\user\Documents\GitHub\Faks FRI\UPK\uporabniku-prilagojena-komunikacij
  dynamic_range[i] = 20*np.log10(data[start:end+1].max()/data[start:end+1].min
MSRS: Lenght :  798000 nsampl :  4320
Temp_stres :  (2320, 4)
Temp_base  :  (4320, 4)
-------- person_feat :  (7080, 25)
++++++++++   data :  (12955, 25)
```

2.3.2 Preglejte izdelane značilke

```
data.describe()
```

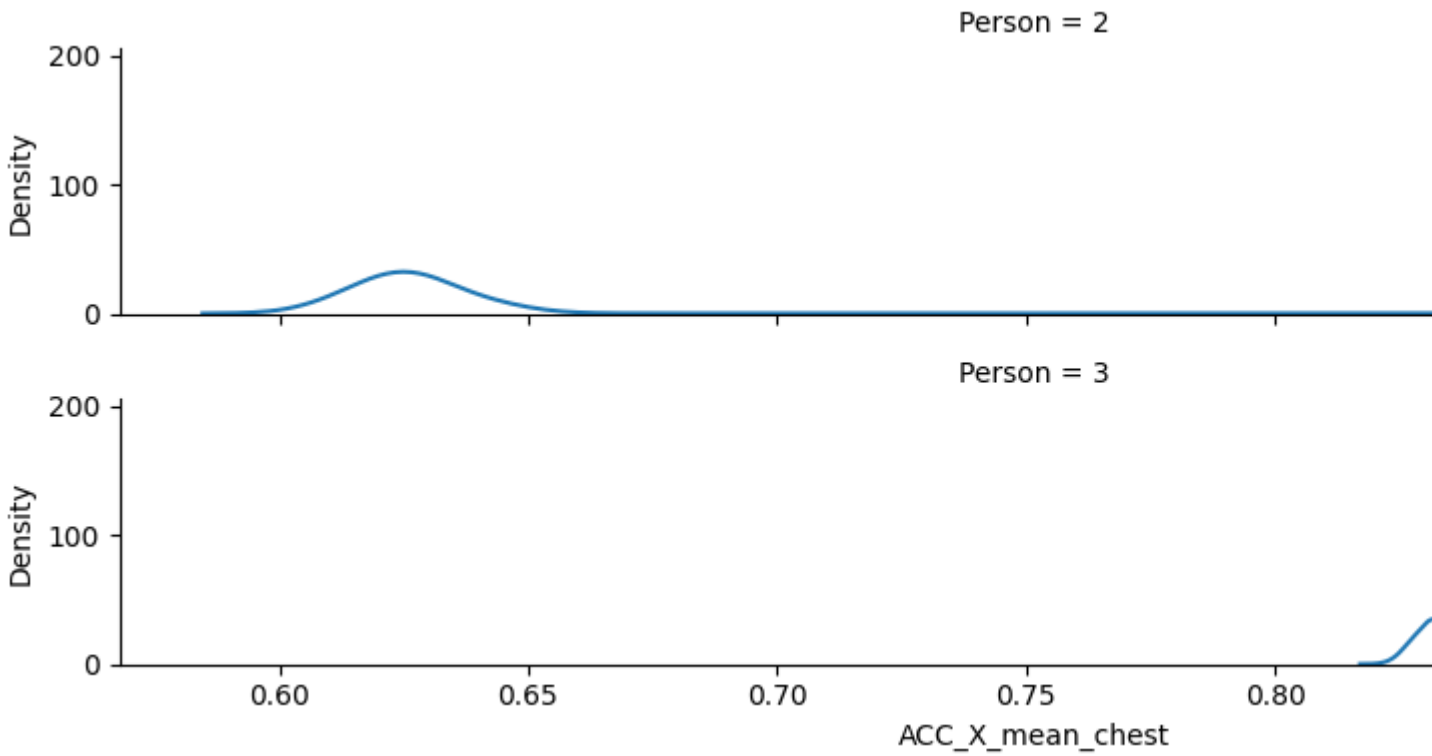
	ACC_X_mean_chest	ACC_Y_mean_chest	ACC_Z_mean_chest	ACC_X_std_ches
count	12955.000000	12955.000000	12955.000000	12955.00000
mean	0.788538	-0.053679	-0.403722	0.00856
std	0.114719	0.057352	0.253848	0.00945
min	0.608975	-0.234158	-0.729365	0.00273
25%	0.630778	-0.088446	-0.708575	0.00445
50%	0.838817	-0.061590	-0.391870	0.00575
75%	0.877610	-0.014675	-0.275681	0.00890
max	0.912743	0.121292	0.102308	0.13645

2.4 Vizualizacija značilk

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

2.4.1 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()
```

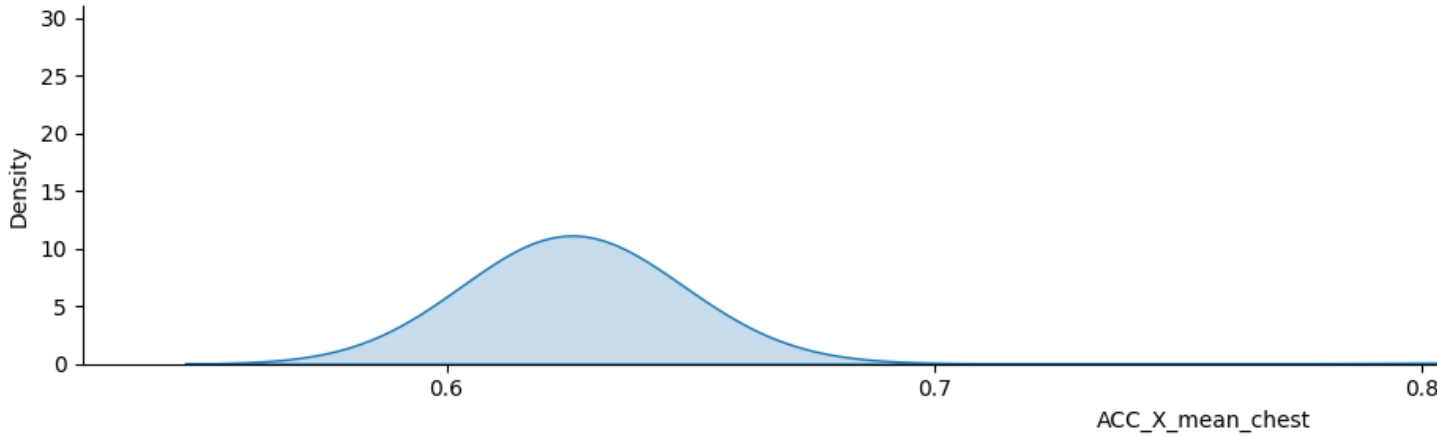


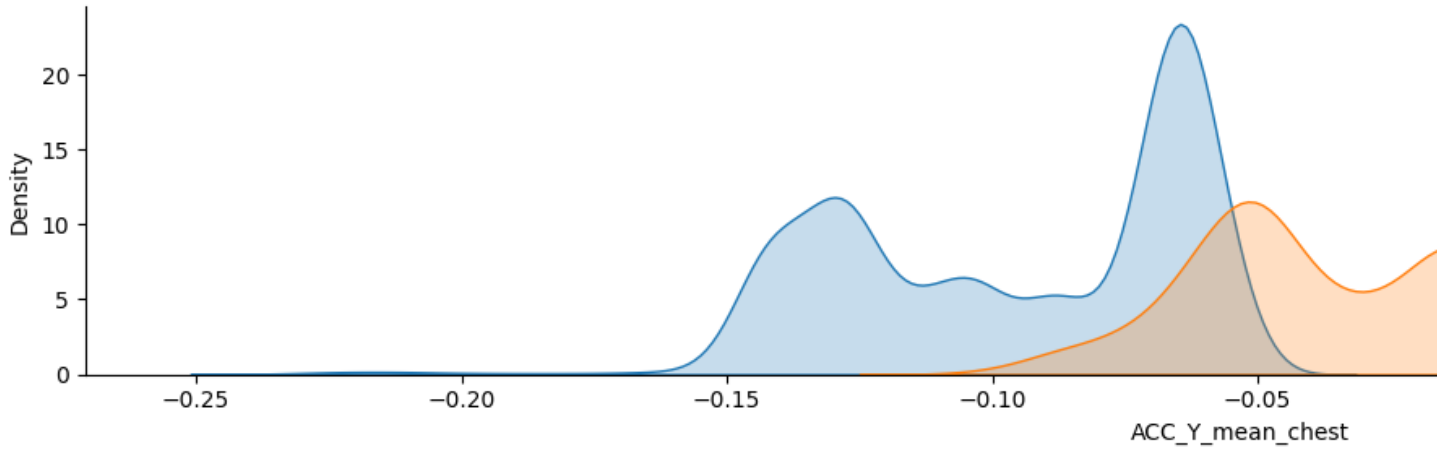
Izriši še za kakšno dodatno značilko. Ali se histogram značilke razlikuje glede na stanje osebe, ali lahko torej iz te značilke ločimo stanji ?

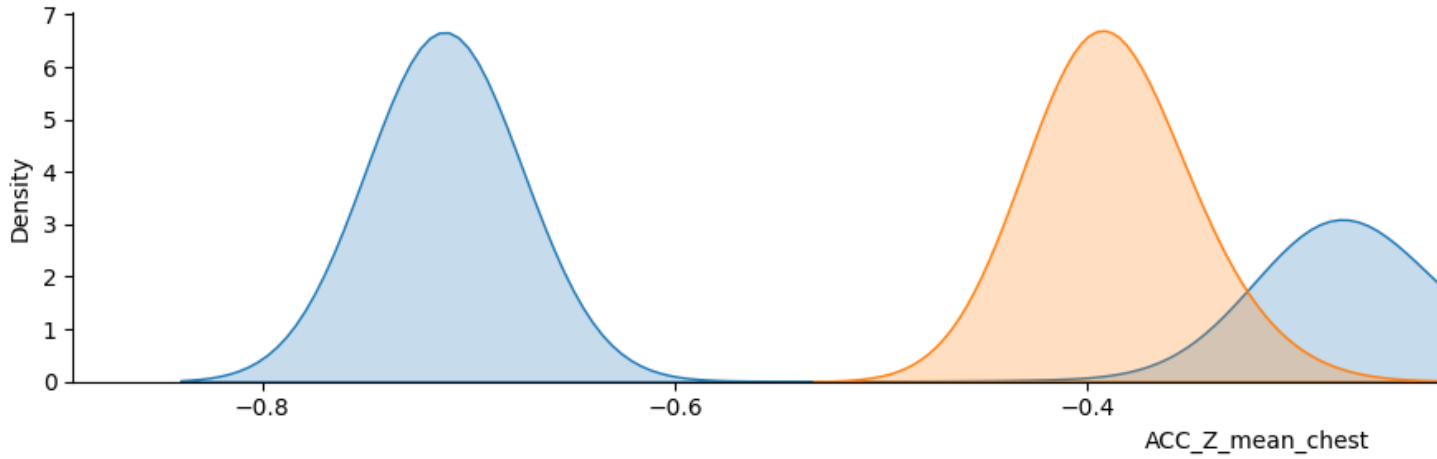
2.4.2 Izris skupne porazdelitve po osebah

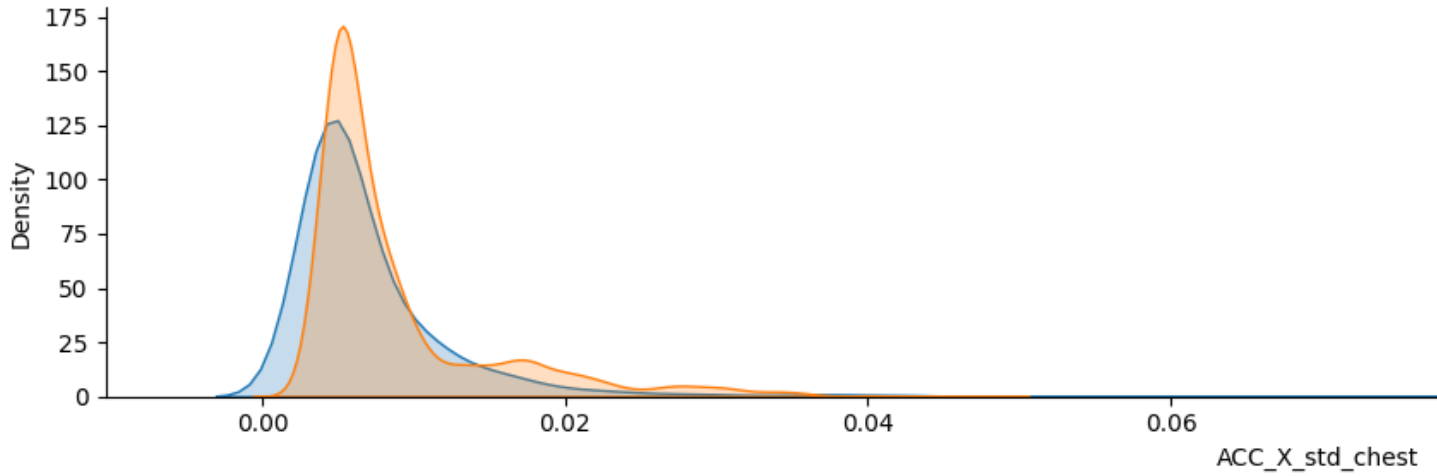
```
import seaborn as sns
import matplotlib.pyplot as plt

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()
```

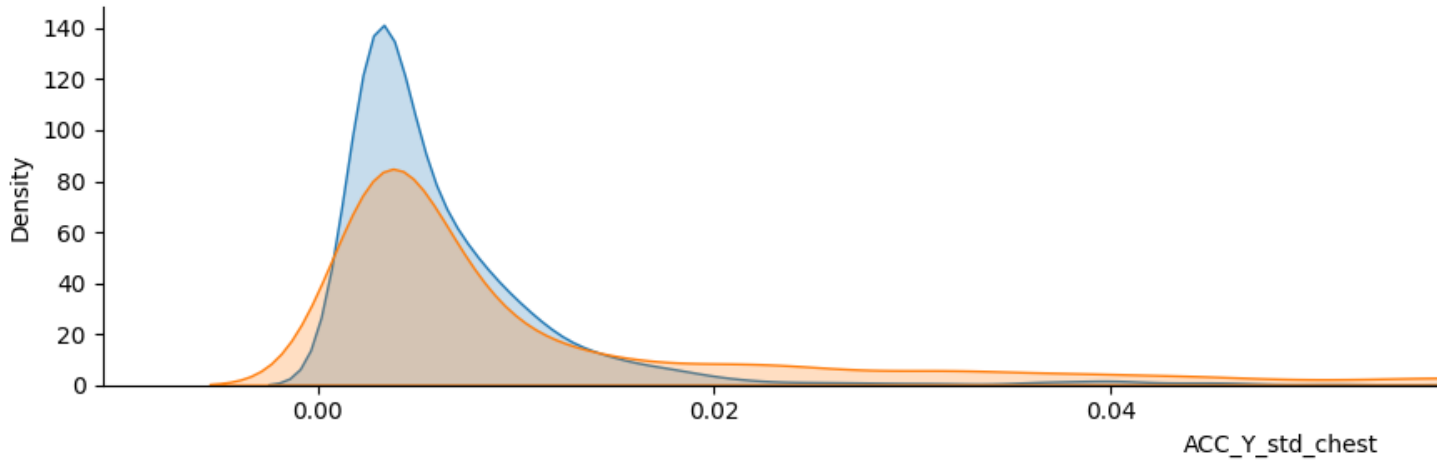
Komentiraj grafe.

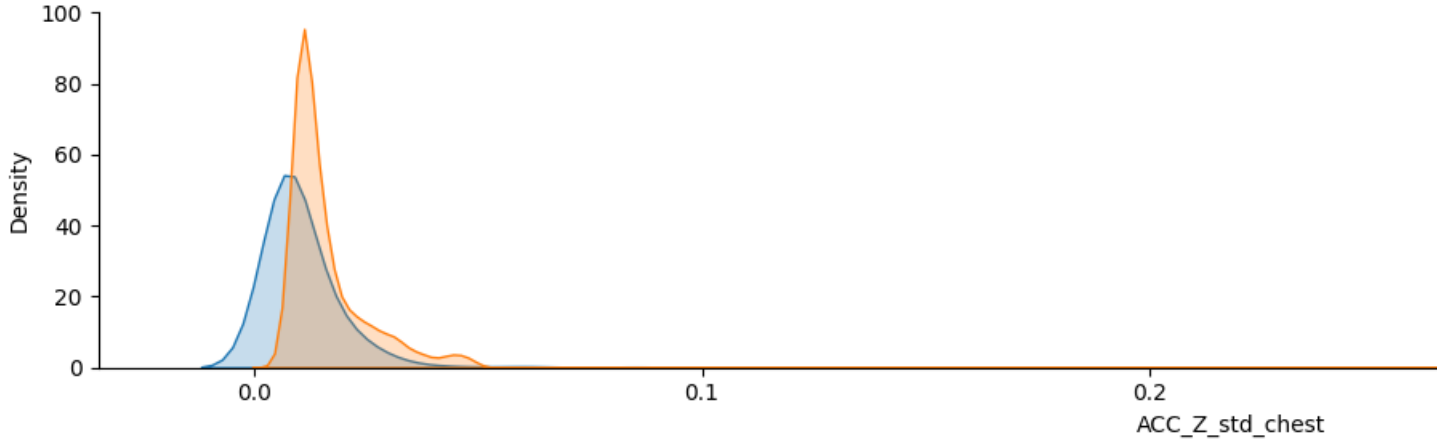

Density
30
25
20
15
10
5
0
0.6
0.7
0.8
ACC_X_mean_chest

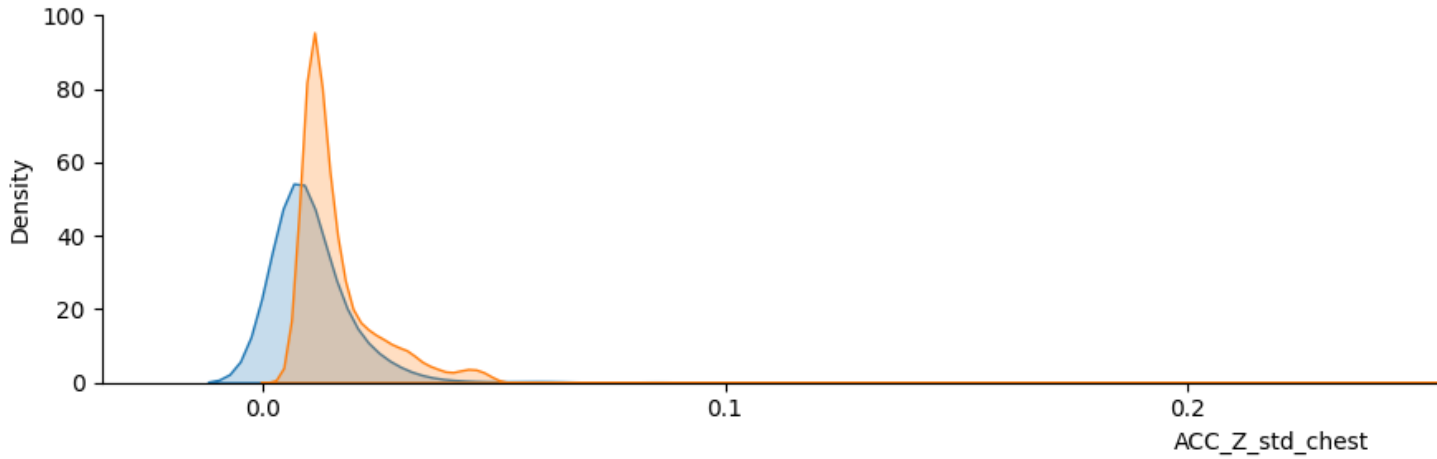

Density
20
15
10
5
0
−0.25
−0.20
−0.15
−0.10
−0.05
ACC_Y_mean_chest

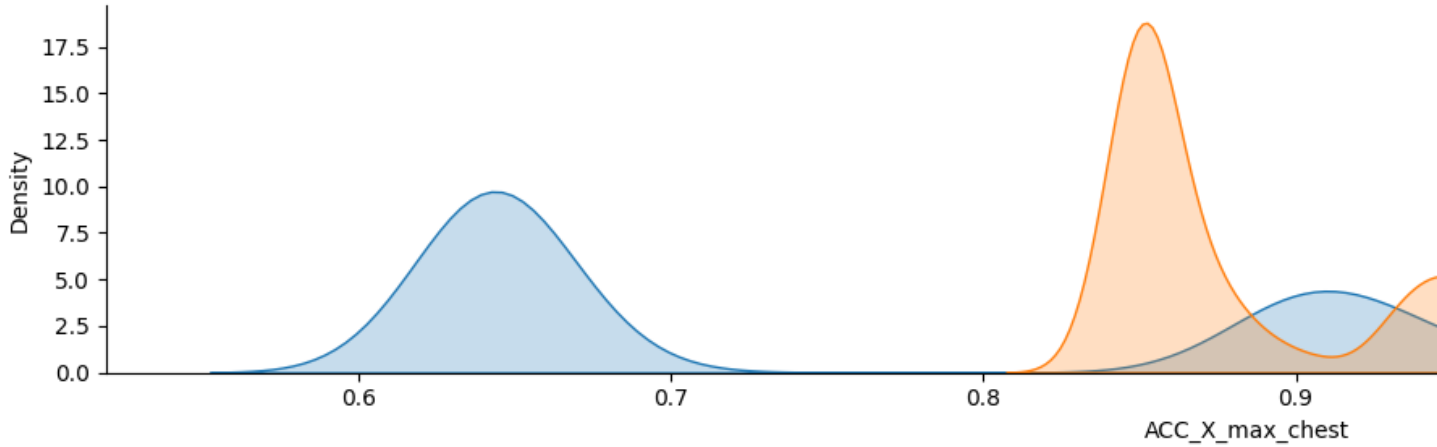

Density
7
6
5
4
3
2
1
0
−0.8
−0.6
−0.4
ACC_Z_mean_chest

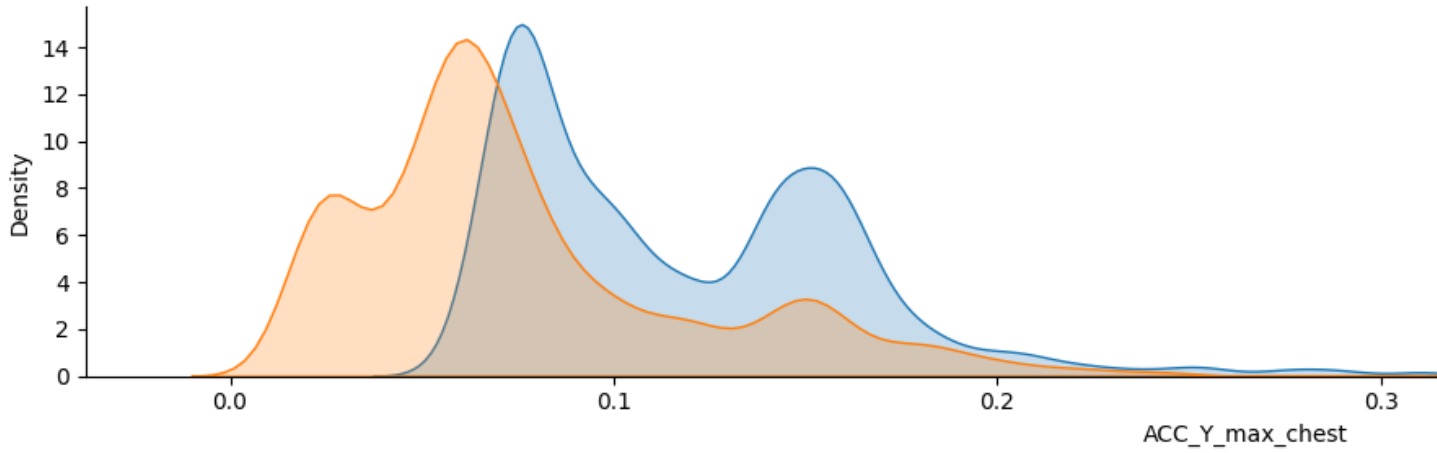

175
150
125
100
75
50
25
0
Density
0.00
0.02
0.04
0.06
ACC_X_std_chest

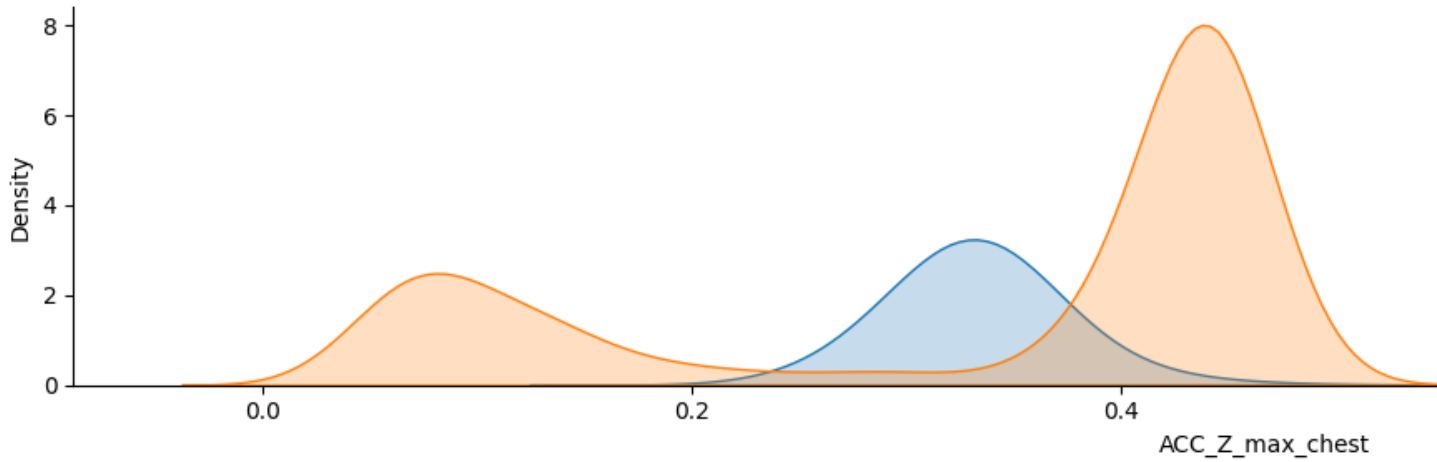

140
120
100
80
60
40
20
0
Density
0.00
0.02
0.04
ACC_Y_std_chest

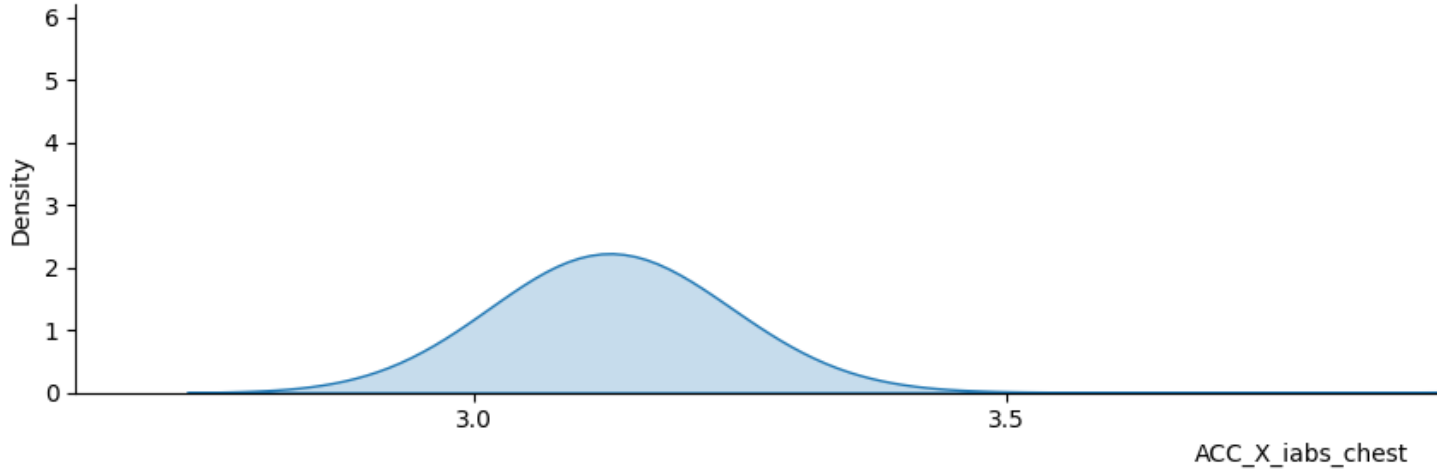

100
80
60
40
20
0
Density
0.0
0.1
0.2
ACC_Z_std_chest


Density
100
80
60
40
20
0
0.0
0.1
0.2
ACC_Z_std_chest


Density
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.6
0.7
0.8
0.9
ACC_X_max_chest


Density
14
12
10
8
6
4
2
0
0.0
0.1
0.2
0.3
ACC_Y_max_chest

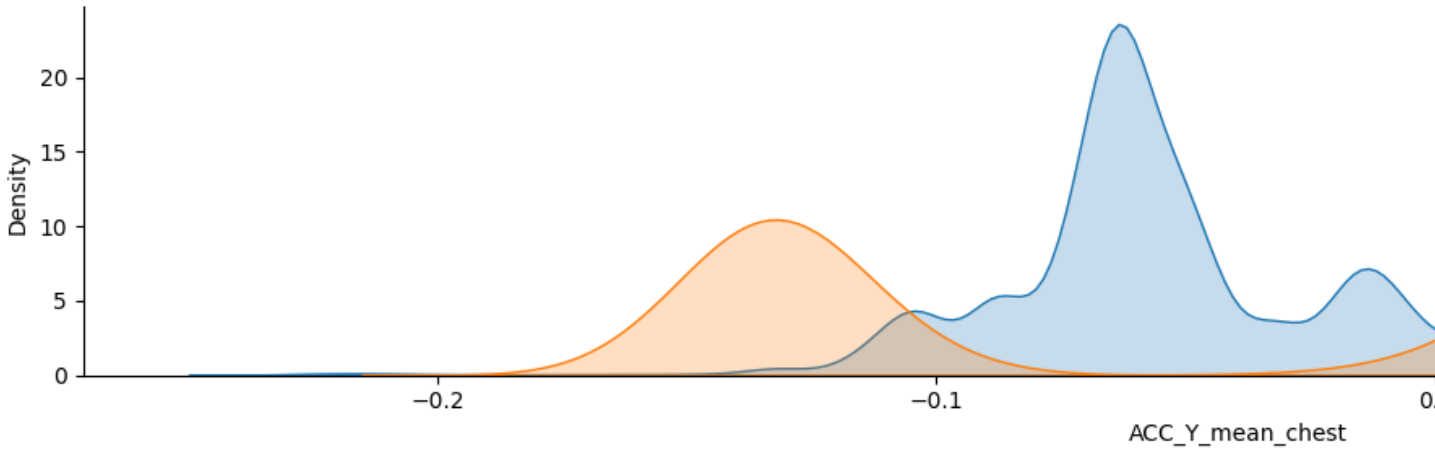


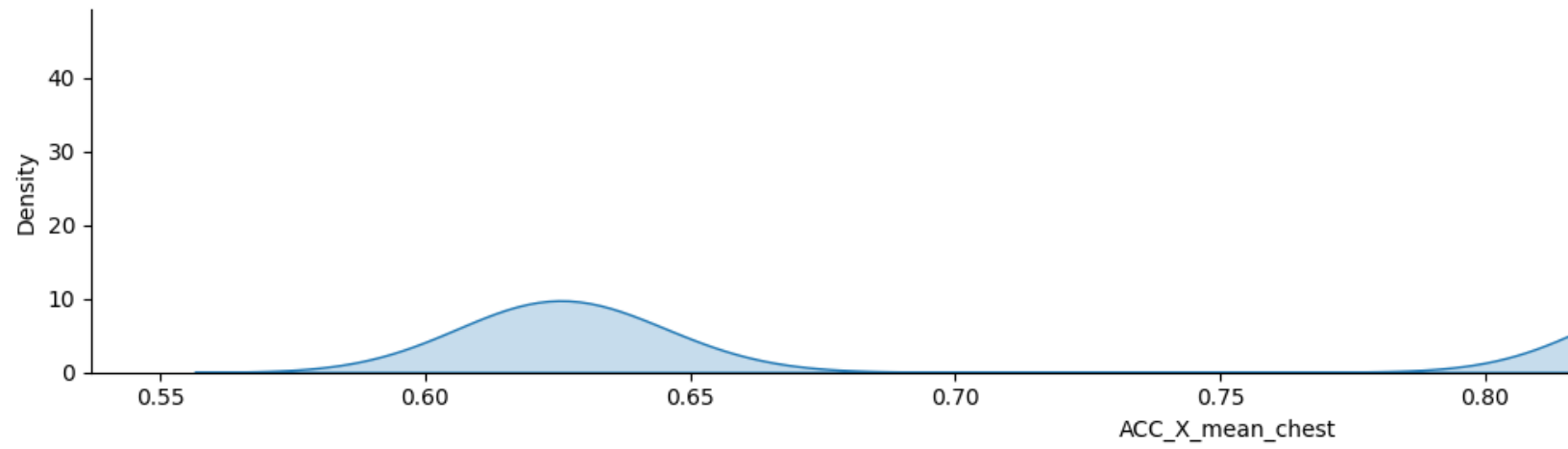


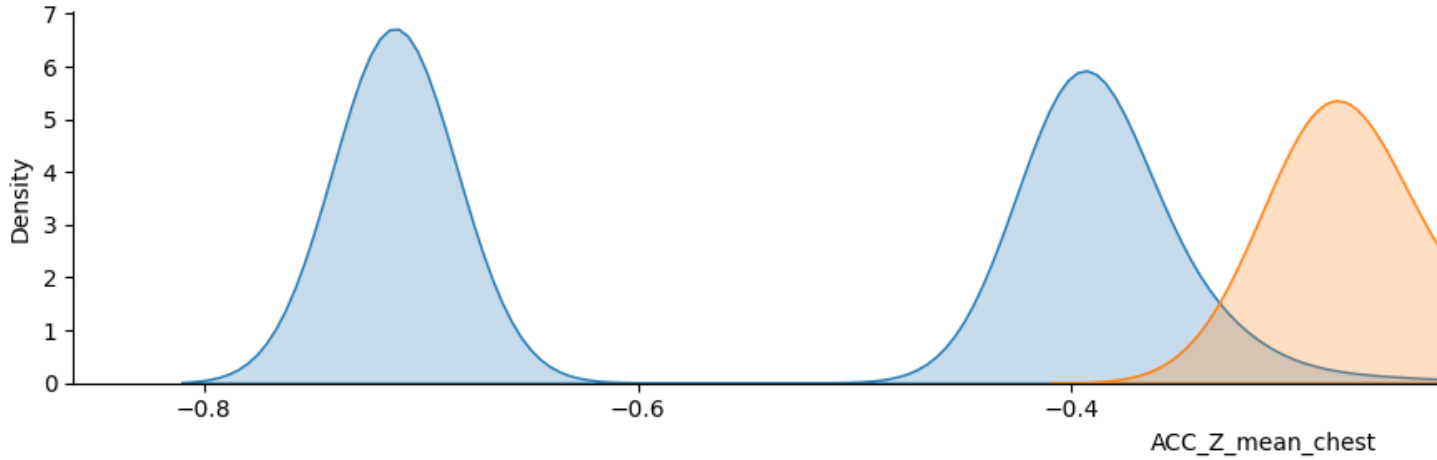
2.4.3 Izris skupne porazdelitve vseh oseb, po stanjih

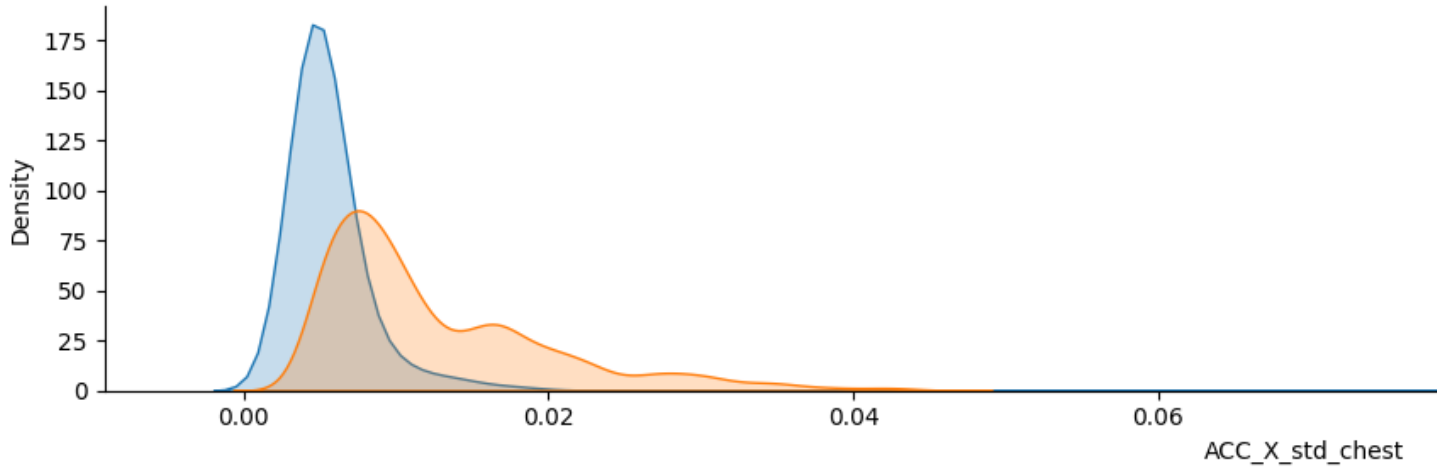
Katere značilke se porazdelitve najbolj razlikujejo za obe stanji ?

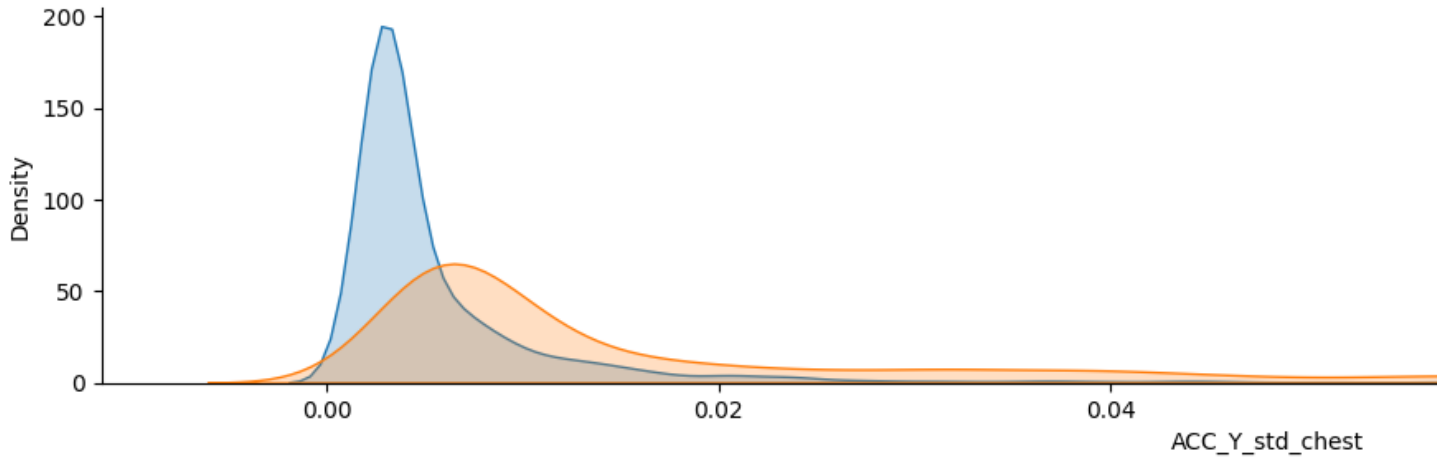
```
import seaborn as sns
import matplotlib.pyplot as plt

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()
```

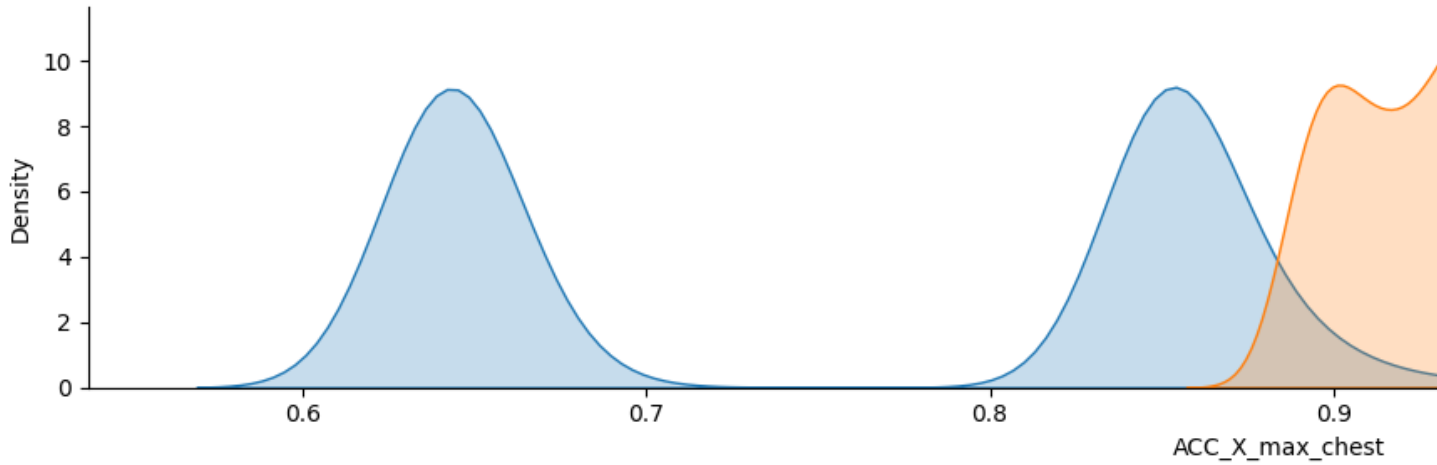

Density
20
15
10
5
0
−0.2
−0.1
ACC_Y_mean_chest

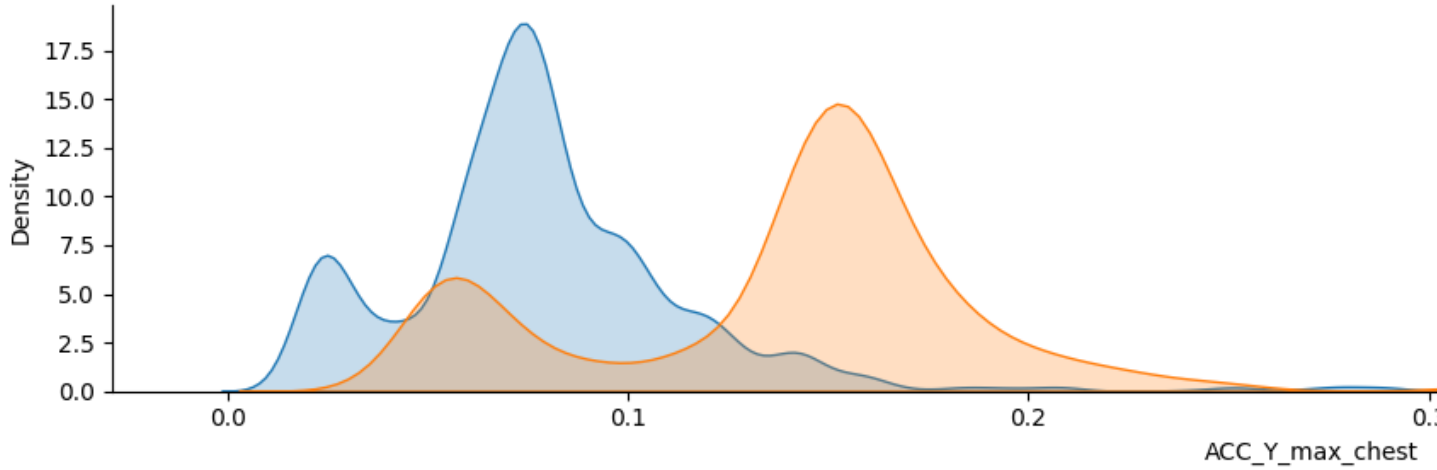

Density
40
30
20
10
0
0.55
0.60
0.65
0.70
0.75
0.80
ACC_X_mean_chest

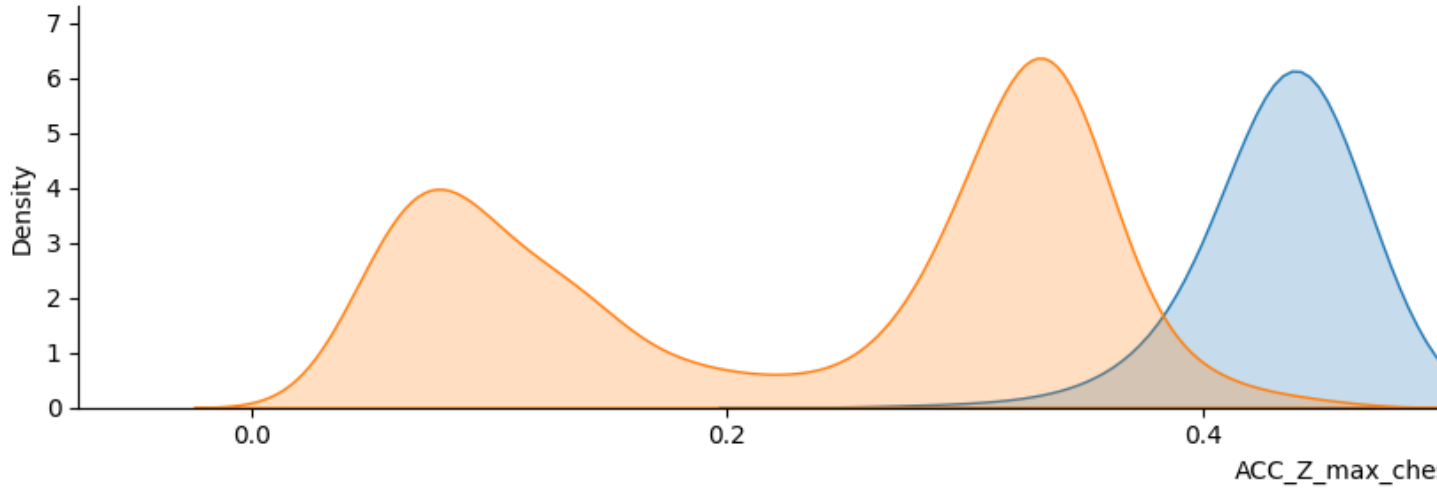

Density
7
6
5
4
3
2
1
0
−0.8
−0.6
−0.4
ACC_Z_mean_chest

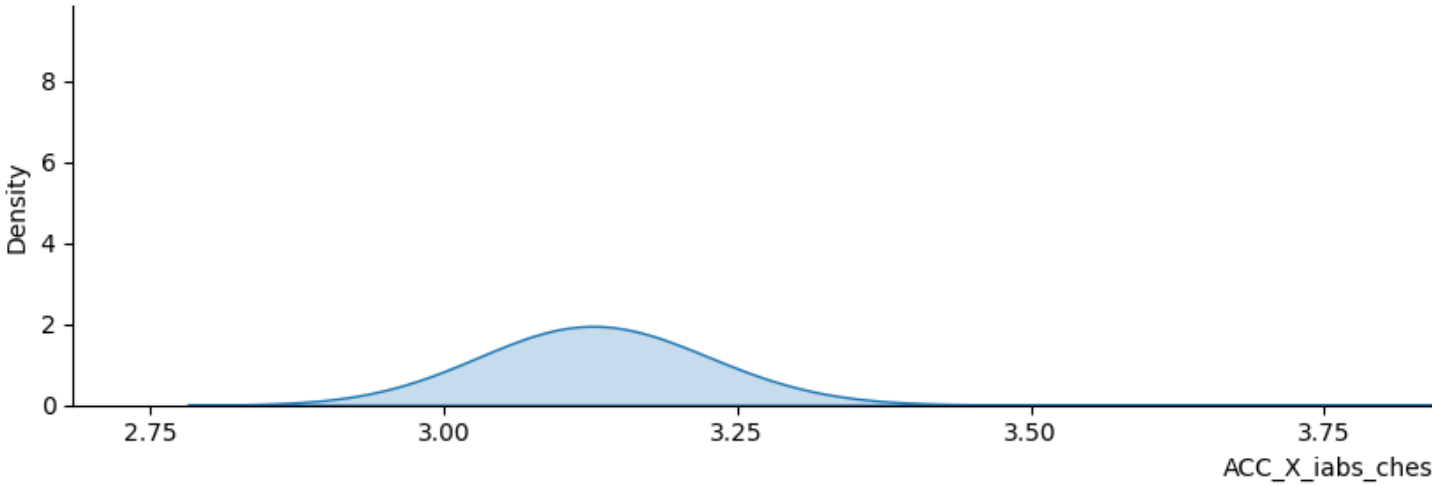

Density
175
150
125
100
75
50
25
0
0.00
0.02
0.04
0.06
ACC_X_std_chest


Density
200
150
100
50
0
0.00
0.02
0.04
ACC_Y_std_chest


Density
10
8
6
4
2
0
0.6
0.7
0.8
0.9
ACC_X_max_chest


Density
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.1
0.2
ACC_Y_max_chest


Density
7
6
5
4
3
2
1
0
0.0
0.2
0.4
ACC_Z_max_che



2.5 Normalizirani podatki

```
### Normalizacija podatkov
norm_data=(data - data.mean())/data.std()

norm_data['Label'] = data['Label']
norm_data['Person'] = data['Person']

norm_data.describe()
```

	ACC_X_mean_chest	ACC_Y_mean_chest	ACC_Z_mean_chest	ACC_X_std_chest	ACC_Y_std_chest	ACC_Z_std_chest	ACC_X
count	1.295500e+04	1.295500e+04	1.295500e+04	1.295500e+04	1.295500e+04	1.295500e+04	1.
mean	-5.265311e-16	-3.510207e-17	-7.020415e-17	-2.632655e-17	1.228573e-16	4.387759e-17	-2
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.
min	-1.565242e+00	-3.146851e+00	-1.282827e+00	-6.163423e-01	-6.092767e-01	-4.318711e-01	-1.
25%	-1.375187e+00	-6.062138e-01	-1.200928e+00	-4.344913e-01	-5.361978e-01	-3.359114e-01	-1.
50%	4.382820e-01	-1.379385e-01	4.668960e-02	-2.978995e-01	-3.907611e-01	-1.811375e-01	2
75%	7.764389e-01	6.800725e-01	5.044002e-01	3.594552e-02	-6.511397e-03	2.884699e-02	7
max	1.082687e+00	3.050789e+00	1.993438e+00	1.352048e+01	7.029478e+00	1.725776e+01	2.

8 rows × 25 columns

Podatki se ne bodo bistven razlikovali ker smo jih normalizirali

Izris porazdelitev:

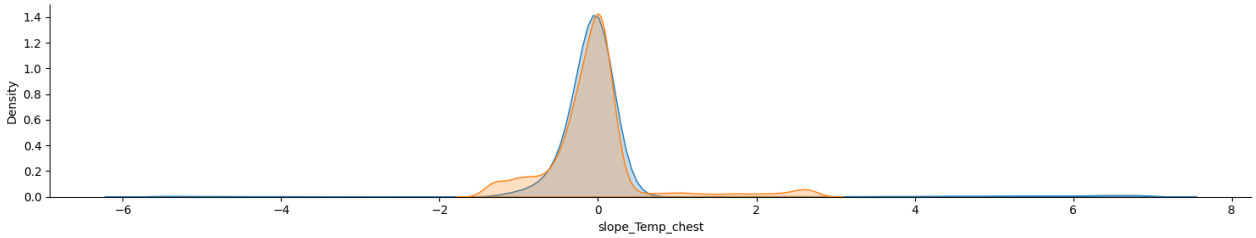
Grafi izgledjo enako ker nismo še med sabo oseb porazdelili.

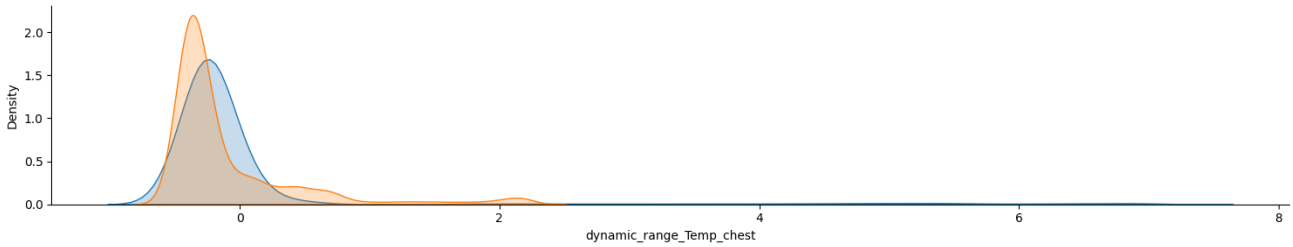
```
for col in norm_data.columns[0:19]:
    fg2 = sns.FacetGrid(norm_data, hue="Label", aspect=5)
```

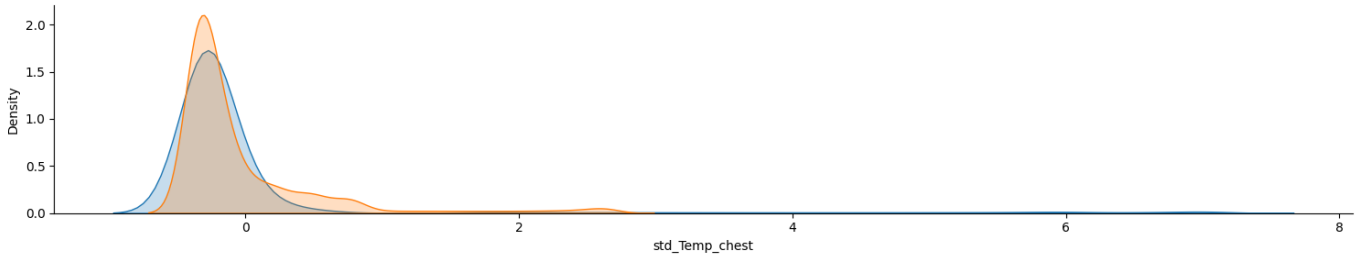
```
    fg2.map(sns.kdeplot, col, fill=True)
plt.show()
```

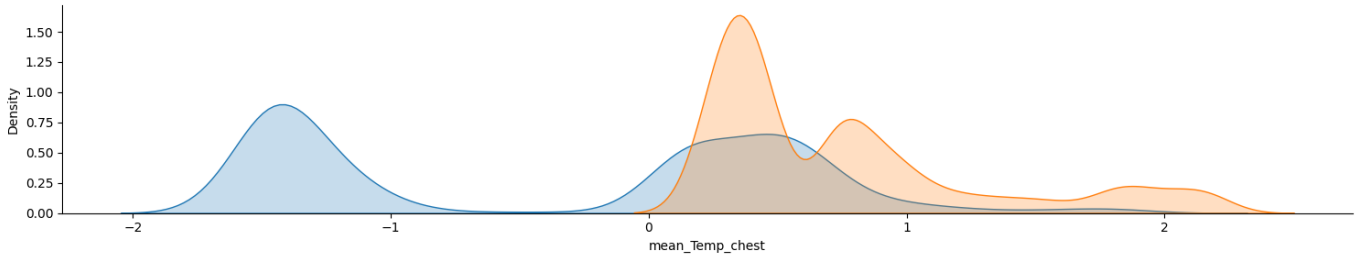
Primerjal sem bazo (modra) in stres (oranžna) stanja. Bolj kot se podatkovne točke prekrivajo, manj primerna je značilka za razlikovanje med tema dvema stanji. Če so podatkovne točke za bazo in stresna stanja tesno skupaj, to pomeni, da značilka ni primerna za zaznavanje razlik med tema stanjema. To 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.









3 Strojno učenje modela prepoznave stanja (stres, bazično)

3.1 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.

3.2 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:

```
 y_train  :  (9068,)
 x_train  :  (9068, 23)
 x_test  :  (3887, 23)
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000      2586
           1     1.0000    1.0000    1.0000      1301

    accuracy                         1.0000      3887
   macro avg     1.0000    1.0000    1.0000      3887
weighted avg     1.0000    1.0000    1.0000      3887
```

Skupna natančnost:

Accuracy: 1.0000

Povprečje (macro avg): 1.0000

Tehtano povprečje (weighted avg): 1.0000

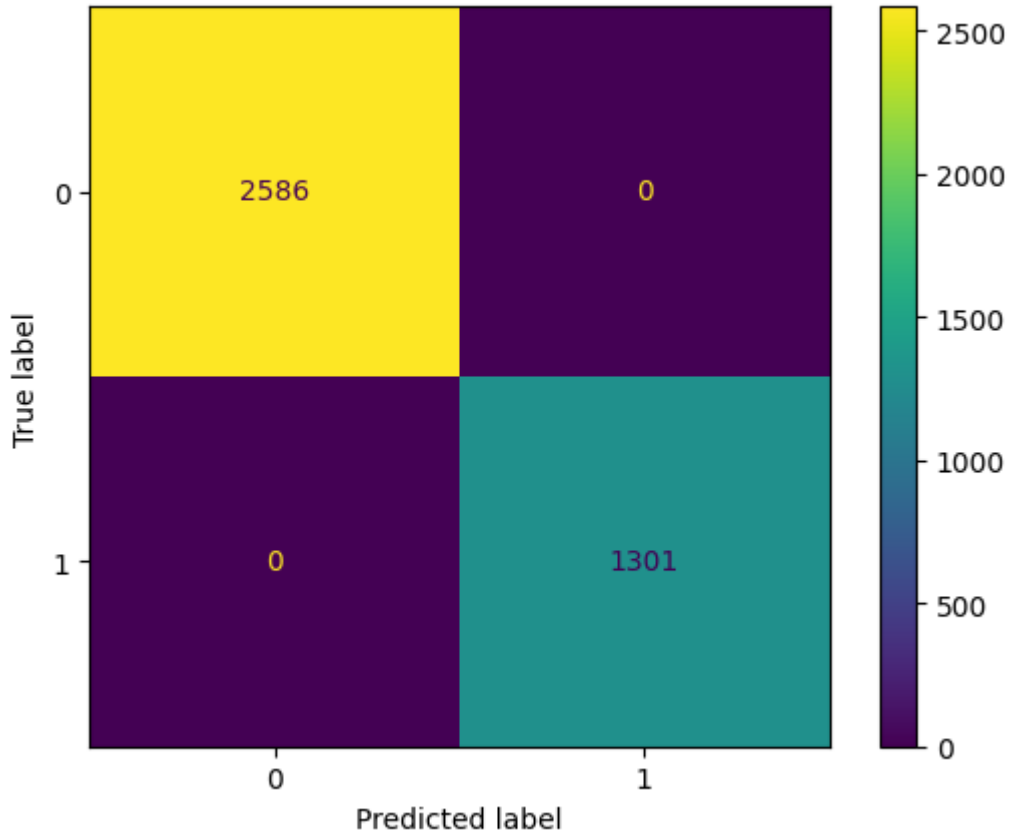
Skupaj testiranih primerov: 3887

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.



3.3 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.

```
0       True
1       True
2       True
3       True
4       True
        ...
2315    True
2316    True
2317    True
2318    True
2319    True
Name: Person, Length: 12955, dtype: bool
<class 'pandas.core.frame.DataFrame'>
Index: 12955 entries, 0 to 2319
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ACC_X_mean_chest         12955 non-null  float64
 1   ACC_Y_mean_chest         12955 non-null  float64
 2   ACC_Z_mean_chest         12955 non-null  float64
 3   ACC_X_std_chest          12955 non-null  float64
 4   ACC_Y_std_chest          12955 non-null  float64
 5   ACC_Z_std_chest          12955 non-null  float64
 6   ACC_X_max_chest          12955 non-null  float64
 7   ACC_Y_max_chest          12955 non-null  float64
...
    accuracy                           1.00      6556
   macro avg       1.00      1.00      1.00      6556
weighted avg       1.00      1.00      1.00      6556
```

- 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!!!**).

```
0       False
1       False
2       False
3       False
4       False
        ...
2335     True
2336     True
2337     True
2338     True
2339     True
Name: Person, Length: 26539, dtype: bool
<class 'pandas.core.frame.DataFrame'>
Index: 19983 entries, 0 to 2339
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ACC_X_mean_chest        19983 non-null  float64
 1   ACC_Y_mean_chest        19983 non-null  float64
 2   ACC_Z_mean_chest        19983 non-null  float64
 3   ACC_X_std_chest         19983 non-null  float64
 4   ACC_Y_std_chest         19983 non-null  float64
 5   ACC_Z_std_chest         19983 non-null  float64
 6   ACC_X_max_chest         19983 non-null  float64
 7   ACC_Y_max_chest         19983 non-null  float64
...
    accuracy                           0.66      6556
   macro avg       0.79      0.50      0.40      6556
weighted avg       0.75      0.66      0.53      6556
```

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.