Příloha 11

Kódové provedení calibration

calibrate_whole_system – algoritmus pro otestování kalibrace systému
apply_calibration – algoritmus pro aplikaci detekčního prahu a získání hodnot (anomálie/neanomálie)
calibrate_threshold – algoritmus pro vytvoření detekčního prahu

---

```
import pandas as pd
import numpy as np
import warnings


def calibrate_whole_system(data, columns, train_size, norm_signal_cat =
"normal_signal", rareness=5):
    # Create testing and training samples
    data_no_anomaly =
data[data['category_name']==norm_signal_cat].reset_index(drop=True)
    data_anomaly =
data[data['category_name']!=norm_signal_cat].reset_index(drop=True)

    reordered_indexes =
np.random.choice(data_no_anomaly.index,int(len(data_no_anomaly)))

    train_data =
data_no_anomaly.loc[reordered_indexes[:int(len(data_no_anomaly)*train_size)
]]
    test_data =
data_no_anomaly.loc[reordered_indexes[int(len(data_no_anomaly)*train_size):
]]
    test_data = pd.concat([test_data, data_anomaly])

    # Create data ready for calibration
    calibration_data = train_data[train_data['category_name'] ==
norm_signal_cat].copy()
    calibration_data.reset_index(inplace=True, drop=True)
    calibration_data = calibration_data.head(20)
    real_calibraton_data = calibration_data.drop(columns=['category_name'])


    # Generate lower and upper threshold
    upper_bandwidth, fp_up = calibrate_threshold(real_calibraton_data,
rareness, 'std_max')
    lower_bandwidth, fp_low = calibrate_threshold(real_calibraton_data,
rareness, 'std_min')

    # Generate data ready for thresholding
    ratio_data = test_data.reset_index(drop=True)
    ratio_data['category'] = 1
    ratio_data.loc[ratio_data['category_name']==norm_signal_cat,
'category'] = 0
    ratio_data.loc[0, 'intensity'] = None
    ratio_data.drop(columns=['category_name'], inplace=True)

    # Threshold it
    ratio_data = apply_calibration(ratio_data, upper_bandwidth,
lower_bandwidth, columns)

    # Main statistics
```

```
    ratio_data.loc[ratio_data['intensity']>0, 'intensity'] = 1
    falsepos_filt = (ratio_data['category'] == 0) &
(ratio_data['intensity'] > 0)
    falseneg_filt = (ratio_data['category'] > 0) & (ratio_data['intensity']
== 0)

    score = sum(ratio_data['intensity'] ==
ratio_data['category'])/len(ratio_data)
    false_positives = sum(falsepos_filt)/len(ratio_data)
    false_negatives = sum(falseneg_filt)/len(ratio_data)

    description =
test_data.iloc[ratio_data.loc[ratio_data['category']!=ratio_data['intensity
']].index]

    returns = {
        'precision': score,
        'false_positive': false_positives,
        'false_negative': false_negatives,
        'test_data': test_data,
        'train_data': train_data,
        'description': description,
        'upper_thresh': upper_bandwidth,
        'lower_thresh': lower_bandwidth
    }

    return returns


def apply_calibration(data, b_up, b_low, columns=list()):
    if not columns:
        columns = b_up.columns

    for row in range(len(data)):
        bool_filt = (data.loc[row, columns] > b_up.loc[:, columns]).loc[0]
| (data.loc[row, columns] < b_low.loc[:, columns]).loc[0]
        int_filt = int(bool_filt.sum())>0
        data.loc[row, 'intensity'] = 1 if int_filt else 0

    return data


def calibrate_threshold(calibration_data, rareness: float = 5, method: str
= 'std_max'):
    """
    Parameters
    -------------
    calibration_data : pandas DataFrame
        Data we want to calibrate on. Contains ONLY NUMERIC values that
        we want to estimate bandwidth on.

    rareness : float
        How rare should be the values of errors. The figher the rareness
        the less false positives values there are but the higher rate of
        false positives.

        Depending on signal noise variation values can be from 5 up to 1e3.

    method : str
        Method of calibration.
        'std_max' - Bandwidth defined by upper standard deviation time
```

```
rareness
        'std_min' - Bandwidth defined by lower standard deviation time
rareness

    Returns
    -------------
    bandwidth : pandas DataFrame
        Bandwidth values with colums corresponding to calibration_data.
        Note: Bandwidth value are calculated for absolute values.

    false_positive : float
        Percentil value of detected false_positive on calibrated signal.
        Must be zero!

    """

    # It is important that we work with absolute values
    calibration_data = calibration_data.applymap(abs)

    data_description = calibration_data.describe()

    # Creation of bandwidth dataframe
    bandwidth = pd.DataFrame(None, columns=data_description.columns,
index=[0])

    if method == 'std_max':
        bandwidth.loc[0] = (data_description.loc['mean',:] +
rareness*data_description.loc['std',:]).values
    elif method == 'std_min':
        bandwidth.loc[0] = (data_description.loc['mean',:] -
rareness*data_description.loc['std',:]).values

    # If calibration was created correctly, then there must be any false
positive
    # detection on tested data.
    false_positive=0

    for row in range(len(calibration_data)):
        if method == 'std_max':
            if (abs(calibration_data.loc[row]) > bandwidth).loc[0].sum() >
0:
                false_positive += 1
        elif method == 'std_min':
            if (abs(calibration_data.loc[row]) < bandwidth).loc[0].sum() >
0:
                false_positive += 1

    if false_positive > 0:
        false_positive = false_positive/len(calibration_data)
        warnings.warn("WARNING: False positive values were found in
calibration signal. Increase rareness value.")

    return bandwidth, false_positive
```