linear_regression.py

```
001 # Main-Script
002 # ##############################################################################
003
004 # This python script automatically launches all other python scripts in the
005 # right order and computes the entire task.
006
007 # Authors:
008 # Christopher Mahn
009 # Silas Teske
010 # Joshua Wolf
011
012 # ##############################################################################
013
014 # Import of Libraries
015 # ------------------------------------------------------------------------------
016
017 import main as settings
018 # import string as st
019 # import random as r
020 # import re
021 # from turtle import position
022 # from scipy import interpolate
023 # from concurrent.futures import process
024 # from turtle import position
025 import numpy as np
026 import math as m
027 # import sys
028 import os
029 # import matplotlib.pyplot as plt
030 # from scipy.fft import fft, fftfreq
031 # from scipy import signal
032 # import multiprocessing as mp
033 # import copy
034 import lib_trajectory as t
035 from sklearn.neural_network import MLPRegressor
036
037
038 # ------------------------------------------------------------------------------
039 # Debugging-Settings
040
041 verbose = True  # Shows more debugging information
042
043
044 # Functions
045 # ------------------------------------------------------------------------------
046
047 def write_text(filename, text):
048     """
049     This function writes text to a file.
050
051     Args:
052         filename (str): Name of the file
053         text (str): Future content of the file.
054     """
055     if(verbose):
056         print(f'[INFO] Writing text to "{filename}"')
057     with open(os.path.join("data", filename), "w") as f:
058         f.write(f'{text}')
059     return(None)
060
061
062 # Classes
063 # ------------------------------------------------------------------------------
064
065
066 # Beginning of the Programm
067 # ------------------------------------------------------------------------------
068
069 if __name__ == '__main__':
070     # Dataset-Information
071     projectnames = settings.project_filenames
072     dataset_length = settings.trajectories_per_project  # Number of datasets per project-name
073     trainingset_length = settings.datasets_per_trajectory  # Number of training-datasets per trajectory
074
075     # Import of individual trajectories
076     for dataset_index, projectname in enumerate(projectnames):
077         for trajectory_index in range(dataset_length):
078             trajectory_ground_truth =
          t.lines_import(f'trajectory_{projectname}_{trajectory_index+1:05d}_ground-truth.csv')  # Ground
          truth data for training
```

```
079
080             # Importing Training-datasets
081             trajectories_measured = []
082             for trainingset_index in range(trainingset_length):
083
        trajectories_measured.append(t.lines_import(f'trajectory_{projectname}_{trajectory_index+1:05d}_t
        raining_{trainingset_index+1:05d}.csv'))
084
085             # Preparing training-data
086             data_groundtruth = []
087             data_measured = []
088             for trainingset_index in range(trainingset_length-1):
089                 for measurement_index, gt_line in enumerate(trajectory_ground_truth):
090                     data_groundtruth.append([gt_line.x1(),
091                                              gt_line.y1(),
092                                              gt_line.x2(),
093                                              gt_line.y2()])
094                     data_measured.append([measurement_index,
095                                           trajectories_measured[trainingset_index]
        [measurement_index].x1(),
096                                           trajectories_measured[trainingset_index]
        [measurement_index].y1(),
097                                           trajectories_measured[trainingset_index]
        [measurement_index].x2(),
098                                           trajectories_measured[trainingset_index]
        [measurement_index].y2(),
099                                           trajectories_measured[trainingset_index]
        [measurement_index].delta_x(),
100                                           trajectories_measured[trainingset_index]
        [measurement_index].delta_y(),
101                                           trajectories_measured[trainingset_index]
        [measurement_index].direction(),
102                                           trajectories_measured[trainingset_index]
        [measurement_index].length()])
103             data_groundtruth = np.array(data_groundtruth)
104             data_measured = np.array(data_measured)
105
106             # Training ML
107             mlpr = MLPRegressor(hidden_layer_sizes=(5, 5), solver='adam', max_iter=50000, verbose=True,
        random_state=1, learning_rate="adaptive", tol=1e-10)
108             mlpr.fit(data_measured, data_groundtruth)
109             write_text(f'result_{projectname}_{trajectory_index+1:05d}_ml-model.txt', mlpr.coefs_)
110
111             # Preparing validation-data
112             validation_data = []
113             for measurement_index, line in enumerate(trajectories_measured[-1]):
114                 validation_data.append([measurement_index,
115                                         line.x1(),
116                                         line.y1(),
117                                         line.x2(),
118                                         line.y2(),
119                                         line.delta_x(),
120                                         line.delta_y(),
121                                         line.direction(),
122                                         line.length()])
123             validation_data = np.array(validation_data)
124
125             # Validating ML
126             ml_prediction = mlpr.predict(validation_data)
127
128             # Export Prediction
129             predicted_trajectory = []
130             for i, e in enumerate(ml_prediction):
131                 predicted_trajectory.append(t.Line(e[0], e[1], e[2], e[3]))
132             t.lines_export(predicted_trajectory, f'result_{projectname}_{trajectory_index+1:05d}_ml-
        prediction.csv')
```