Problem 3: Maneuver Detection for the Loft Dynamics Helicopter Simulator

1. Description

The partner of this project is the Dübendorf-based company Loft Dynamics, who develops and produces the world's only EASA- and FAA-qualified VR helicopter pilot training simulators.

During a training session on the simulator, a helicopter pilot flies a series of different maneuvers, such as hovering, climbs, descents or turns. To analyze maneuvers in more detail – either manually by an instructor, or automatically using a software – all maneuvers during a flight have to be detected. So far, Loft Dynamics uses a hand-crafted rule-based algorithm to detect these maneuvers. However, this solution has a series of downsides: it is difficult to extend these rules to new aircraft, new maneuvers, or fix problems in the detection. In this project, your task is to develop a machine learning (ML) based algorithm which solves these issues by learning from past data of maneuvers.

You should follow the steps outlined below:

1. Develop code for reading the given data and visualize the results. Get an overview of the content and amounts of data, as well as the data format.
   Hint: select a small subset of the inputs for the next steps to keep dimensionality low. You can extend this set later on.
2. Develop a pre-processing method to convert the data into a format which is usable for machine learning. This includes dividing the given signals into blocks of a constant size and replacing the raw values with high-level features. See the appendix of this document for more information.
3. Design and implement a machine learning algorithm to classify the dataset created with your pre-processing method into the different maneuvers.
4. Evaluate and document (!) the accuracy of your ML system, visualize the results, analyze and interpret these results.
5. Iterate the previous steps to improve the accuracy of your system. Your goal is to be the team with the best algorithm.

Note: There will be an award for the team with the best algorithm! The award does not automatically go to the team with the highest accuracy, but an overall best or most interesting solution as judged by the Loft Dynamics team.

2. Formal requirements

1. This project is a group project. Create a group of 3 – 4 students to work on this project together and submit the names of all team members on Moodle until **28 October 2024 at 23:59**.
2. The main deliverable for this project is an oral presentation of 3-5 minutes per team. The presentation must include the following:
   - Short description of your approach (pre-processing, algorithms, settings).
   - Achieved results (including Maneuver-Recall value and other results as judged important by you).

   The presentations will take place **2 December 2024** during the regular lab sessions.
3. Additionally, every team must submit their results and their code on Moodle until **2 December 2024 at 07:59** in the morning. The following content needs to be handed in:
   - Slides for your presentation in PDF format.
   - ZIP file which contains all code required to reproduce the results.
   - A file that defines a reproducible environment for running the code, e.g. a Dockerfile, requirements.txt, setup.py, pyproject.toml or similar.
   - The Maneuver-Recall value on the test data.

Note: There is a question forum on Moodle, where all questions on the project – both technical and administrative – can be asked. In addition, you can always ask questions during the lab sessions.

3. Submission & deadlines

Deadline for submitting teams:	28 October 2024 at 23:59 Submission on Moodle | Week 7 | Submission of Team Members
Deadline for submitting results:	02 December 2024 at 07:59 Submission on Moodle | Week 12 | Submission of Results
Date of presentation:	02 December 2024 during the regular lab sessions You will be assigned a lab session, during which you need to present. At least 2 team members need to be present during the lab session.
Final event & award ceremony	02 December 2024 from 17:00-18:00 The award will be presented in a short final event for all participants. To receive the award, at least 1 team member needs to be present in this final event and present their solution to all. Of course there will be snacks & drinks!

4. Evaluation

Your solution is marked with “pass” or “fail”. It is graded as “pass”, if

- Your solution was submitted on time.
- At least **7 out of 10 points** in the following evaluation is reached:

Criteria	0 Points	1 Point	2 Points	Comment
1. Pre-processing method (including feature engineering) is designed and implemented appropriately.	☐	☐	☐	
2. Machine learning model is implemented correctly.	☐	☐	☐	
3. Evaluation of the performance is done correctly and interpreted reasonably.	☐	☐	☐	
4. Presentation contains the relevant information.	☐	☐	☐	
5. Formal requirements are met and submitted in time.	☐	☐	☐	
Total points:	___ / 10 Points			
Pass / Fail:	☐ Pass ☐ Fail			

Appendix: Brief introduction to time series processing

In general machine learning, we assume that our input X and output Y are given as a table, as shown below:

Index	X_1	X_2	X_3	…	X_p	Y
0						
1						
2						
3						
…						
n						

Each row is one data point, X_1 to X_p are the inputs and Y is the corresponding output. This makes sense for data where you e.g. collect measurements for one person (X_1 to X_p) and try to predict the corresponding output Y for that person. In time-series data, we have a similar structure, however the Index is the time. This leads to 2 problems:

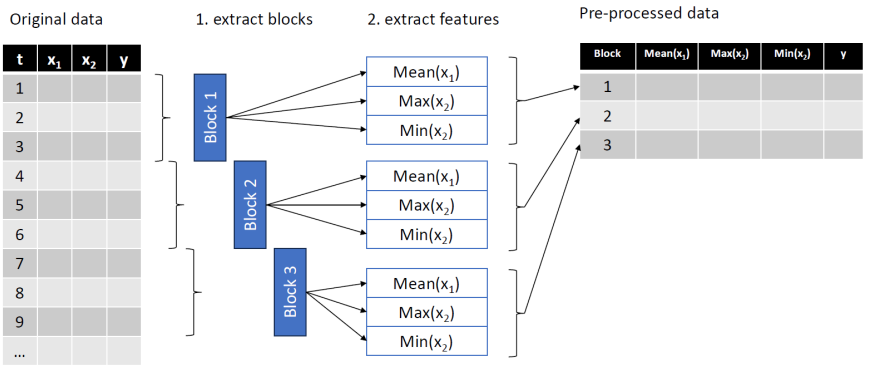
1. If we build a machine learning model on this, only one time instant is available to decide. The recent history (last time steps) are not used and treated individually.
2. As discussed in lecture 4 (linear regression), the error terms will usually be highly correlated between the time instant t and t+1. This violates our fundamental assumptions and will often lead to a bad performance.

To solve these two issues, the following two steps are usually done as a pre-processing for time-series data:

1. The data is divided into blocks containing multiple time steps. The block size is a parameter you must find yourself. You want to make it small enough, so you don't have long-term drifts (e.g. the helicopter climbing from ground level to its cruise altitude), however large enough to capture structure that is relevant for the prediction (e.g. the fact that you are indeed climbing with a measurable climbing rate).
   Note: Blocks can be (and very often are!) overlapping.
2. Within such a block, the data is usually highly correlated and thus redundant. Thus, the raw measurements are replaced by high-level features which characterize the data within the block. For example, the average altitude within a block might be a useful feature. Typical features are:
   - Average, Standard deviation
   - Minimum, Median, Maximum
   - Skewness, Kurtosis
   - Frequency-domain features

   This "feature engineering" process is one of the most important tasks in time-series processing.

These two steps are visualized in the figure below. On this new, pre-processed dataset, any machine learning method can be applied.



Appendix: Description of the Dataset

The dataset contains measurements from 19 different test flights. For each test flight, two files are available:

- A *.parquet file (Parquet file) which contains all logged data.
  Hint: use the pandas.read_parquet() function to load the data. This function requires the pyarrow library.
- A *.json file which contains the ground truth labels, i.e. the maneuvers as specified by the instructor during the test flight.

In addition, a file *StateDescriptions.json* is given, which contains the mapping from *StateId* (which is the column name in the Pandas dataframe) to human-readable state names which will be helpful to analyze the data.

You need to split the data into a training dataset and a test dataset. You must use the following datasets as test dataset, hence, they may not be used for training or validation purposes. All other datasets may be used for training.

- 0b3f3902-2c04-4625-8576-3bb963e3d709
- 663f573a-74c5-4368-b60b-1fb433cd835d
- 8c36586f-94e9-4ae9-8384-0f3342008677
- a376807a-82d3-4526-b19f-98d4b3f9078b
- d76bb0eb-bc08-4b35-8c1f-37369452083d
- f40f71de-5cc2-4719-8a5a-abcf950cbd71

Below, you will find some hints regarding the data:

- The height of the aircraft above ground is very helpful to visualize and get an overview of the flight.
- The ground truth labels only contain the maneuvers. In addition, there obviously needs to be a class “No Maneuver”, for which you have to generate the labels.
- The labels were collected by human instructors. All humans make errors – do not always trust these labels blindly.
- A *NaN* or *None* in the data means that the value has not changed since the last valid measurement.
- The datasets use the pandas MultiIndex feature, as it has two index variables: TimeStamp and FrameCounter, where the timestamp contains the exact date and time of that measurement, while FrameCounter is an integer counter, starting at 0 in each test flight.

A metric that is important for Loft Dynamics to compare the results, is the Maneuver-Recall value. It is defined as follows: A maneuver is counted as “detected” if during 80% of the (ground-truth) duration of the maneuver, the output of the machine learning model is correct. For example, if a maneuver takes 20 seconds according to the ground truth labels, it has to be detected during at least 16 seconds. If this is the case, this counts as one detected maneuver. The Maneuver-Recall is then given by counting, how many of all maneuvers are detected correctly.

$$ManeuverRecall = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M_i} I(m_{i,j} = detected)}{\sum_{i=1}^{N} M_i}$$

Where N is the number of data files, M_i is the number of maneuvers in file number i, and $I(m_{i,j}=detected)$ is an indicator function which is 1 if the j-th maneuver in the i-th file is detected correctly and 0 otherwise. The normalization $\sum_{i=1}^{N} M_i$ counts the number of all maneuvers in the data. In addition, the Maneuver-Recall for individual maneuvers (e.g. only for the Hover maneuver) are of interest.