Learning from Trajectory Data with MobiML

Anita Graser[1,*], Melitta Dragaschnig[1]

[1]AIT Austrian Institute of Technology, Vienna, Austria

Abstract
This demo paper introduces MobiML, a new library that aims to help scientists and engineers with developing mobility ML solutions using trajectory data. We also demonstrate how MobiML can speed up ML development workflows using the example of a reproduction of the workflow for training a GeoTrackNet trajectory anomaly detection model. MobiML is available at: https://github.com/movingpandas/mobiml.

Keywords
Machine learning, GeoAI, Mobility Data Science

1. Introduction

The Mobility Data Science ecosystem is incredibly diverse with numerous heterogeneous data sources and use cases. Common machine learning (ML) use cases include [1]: location classification, arrival time prediction, traffic volume / crowd flow prediction, trajectory prediction, (sub)trajectory classification, next location / destination prediction, anomaly detection, and synthetic data generation.

A major hurdle in Mobility Data Science "is that existing ML and analytics tools [...] do not support location and mobility as base data types to reason about. [...] This raises a fundamental and big question on what are the analysis primitives and common building blocks for applications that could shape a framework of ML-based mobility data analysis?" [2]

To fill this gap, we present MobiML, a framework for learning from movement data that proposes essential building blocks to build ML solutions based on movement trajectory data. The following section provides information on related libraries. Then we present MobiML and its components, before we present usage examples, and finally present perspectives on future developments.

2. Related Work

To the best of our knowledge, existing ML libraries for spatiotemporal data focus on remote sensing imagery and similar gridded datasets:

GeoTorchAI [3] is a spatiotemporal deep learning framework on top of PyTorch and Apache Sedona. It enables spatiotemporal machine learning practitioners to easily and efficiently implement deep learning models targeting the applications of raster imagery datasets and spatiotemporal non-imagery datasets.

TorchGeo [4] is a PyTorch library by Microsoft that is similar to torchvision and provides datasets, samplers, transforms, and pretrained models specific to geospatial data for remote sensing imagery data.

In the development of MobiML, we have taken inspiration from both GeoTorchAI and TorchGeo and adapted the concepts to trajectory data.

For trajectory data processing, MobiML leverages MovingPandas [5, 6] and therefore, by extension, Pandas and GeoPandas [7], as well as PyMEOS [8] extending MEOS [9].

3. MobiML

The goal of MobiML is to enable the efficient development of ML solutions by providing mobility-aware building blocks for ML workflows. This means that the MobiML tools are aware of the spatiotemporal nature of trajectory data and automatically account for them. MobiML includes the following components: datasets, preprocessing tools, samplers, transformation tools, and models, which are shown in Figure 1 and described in more detail in the following.

3.1. Datasets

This module contains classes for handling popular movement datasets. These classes serve to facilitate model development by providing straightforward access to common public datasets and also serve as templates for custom datasets classes that developers may want to create. Dataset classes provide a standardized interface including functions to access the data in the form of Pandas DataFrames, GeoPandas GeoDataFrames, and MovingPandas TrajectoryCollections, as well as to create static and interactive plots of the data.

Every class provides information on where to get access to the respective dataset (where to download it). The loading function then takes care of the multitude of different input formats, including pickle, csv, feather, zipped csv files, or geo data formats supported by GeoPandas such as Shapefile, GeoPackage, or GeoJSON and maps the spatiotemporal information and mover IDs to the standardized structure.

So far, we have implemented six dataset classes based on real-world public mobility datasets from the:

- Maritime domain: AIS data from Denmark `AISDK` and France `BrestAIS`,
- Urban domain: tracks of cyclists `CopenhagenCyclists`, buses `DelhiAirPollution`, and taxis `PortoTaxis`, and
- Ecology domain: tracks of migratory birds `MovebankGulls`.

3.2. Preprocessing

This module contains tools to preprocess movement data to generate datasets that are ready for ML development.

Published in the Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference (March 25-28, 2025), Barcelona, Spain.
*Corresponding author.
anita.graser@ait.ac.at (A. Graser); melitta.dragaschnig@ait.ac.at (M. Dragaschnig)
0000-0001-5361-2885 (A. Graser); 0000-0001-5100-2717 (M. Dragaschnig)

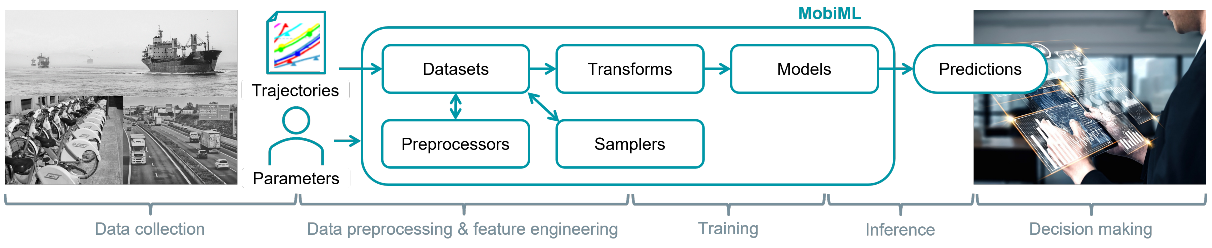

Figure 1: MobiML component overview.

Preprocessing tools always return a mobiml.Dataset object. The developed preprocessing tools include:

- `TrajectoryDownsampler` to reduce the number of points in a trajectory to a certain target sampling interval.
- `TrajectoryEnricher` to add features such as speed, direction, and acceleration in a variety of supported units.
- `TrajectoryFilter` to remove trajectories based on a their number of points or their speed.
- `TrajectorySplitter` to split long trajectories into shorter subtrajectories, for example, based on observation gaps
- `Normalizer` to min-max normalize latitude, longitude, speed, and direction values in dataset.
- `StationaryClientExtractor` to extract subsets of the data based on static locations (provided as GeoDataFrame).
- `MobileClientExtractor` to extract subsets of the data based on moving locations (provided as MobiML Dataset) using PyMEOS.

3.3. Samplers

This module contains tools for sampling movement data while accounting for its spatiotemporal characteristics. The developed sampling tools include:

- `MoverSplitter` to split a dataset ensuring that trajectories of a given percentage of the movers are assigned to the test set. The remaining mover trajectories are assigned to the train set.
- `RandomTrajSampler` to randomly sample trajectories, targeting an equal spatial distribution based on user-defined grid (i.e. equal number of trajectories per cell, based on start points), inspired by [4].
- `TemporalSplitter` to split dataset temporally into training, development, and test sets, ensuring that a given percentage of the time is assigned to each set.

3.4. Transforms

This module contains various transformation operations that can be applied to datasets. Transforms convert a mobiml.Dataset into a different data structure that conforms to model-specific requirements. The developed transformation tools include:

- `DeltaDatasetCreator` to convert absolute locations and timestamps into relative changes, based on [10].
- `ODAggregator` to extract start and end points (OD) for trajectories from a Dataset and aggregate them in a hexagonal (H3) grid.
- `TrajectoryAggregator` to create summary features describing trajectories.

3.5. Models

This module contains example models used to demonstrate mobility ML workflows. Included are:

- GeoTrackNet: Anomaly detection in maritime traffic patterns, as presented in [11].
- Nautilus: Vessel Route Forecasting (VRF), as presented in [10].
- SummarizedAISTrajectoryClassifier: an example model for trajectory classification in a federated learning setting.

The use of these components is documented through a series of example notebooks that demonstrate individual components as well as ML workflows from data preprocessing to model training and inferencing.

4. Demo

In this section, we present how MobiML can be used to train GeoTrackNet using Danish AIS data[1]. The original GeoTrackNet paper [11] uses AIS data from a different source. Our example demonstrates how to use GeoTrackNet with Danish AIS data. This example workflow makes use of multiple MobiML components from datasets to preprocessing, samplers, and of course the model itself.

4.1. Step 1: Data Loading

Using MobiML's `AISDK` dataset class, we can easily load one of the CSV files provided by the Danish Maritime Authority, as shown in Figure 2.

4.2. Step 2: Preprocessing

To preprocess the data, we leverage classic Pandas DataFrame operations and MobiML tools as follows:

1. Filter the dataset to our desired vessel types by directly working with the AISDK object's DataFrame (Figure 3),
2. Split trajectories at observation gaps using `TrajectorySplitter` (Figure 4) since these gaps would mess up what the model learns,

[1]for the full notebook see https://github.com/movingpandas/mobiml/blob/main/examples/mobiml-geotracknet.ipynb

```
path = os.path.join(data_path, csv_filename)
print(f"{datetime.now()} Loading data from {path}")
aisdk = AISDK(path)  # you can specify a bounding box here to filter the area
LON_MIN, LAT_MIN, LON_MAX, LAT_MAX = aisdk.get_bounds()
print(
    f"Bounding box:\nmin_lon: {LON_MIN}\nmin_lat: {LAT_MIN}\nmax_lon: {LON_MAX}\
)
```

```
2024-10-22 22:10:30.785165 Loading data from data/aisdk_20180208_sample/aisdk_2018
2024-10-22 22:10:32.524538 Loaded Dataframe with 280849 rows.
Bounding box:
min_lon: 11.3
min_lat: 57.4
max_lon: 12.013717
max_lat: 57.9
```

Figure 2: Loading AIS data using MobiML's AISDK dataset class.

3. Drop trajectories that have too few points using `TrajectoryFilter` (Figure 5), and finally
4. Reduce the dataset size using `TrajectoryDownsampler` (Figure 6).

Since every preprocessing tool takes a Dataset as input and produces a Dataset as output, they can be chained in a modular way to support different needs and workflows.

```
aisdk.df = aisdk.df[
    (aisdk.df["ship_type"] == "Cargo")
    | (aisdk.df["ship_type"] == "Tanker")
    | (aisdk.df["ship_type"] == "Passenger")
]
print("After keeping only 'Cargo', 'Tanker' or 'Passenger' AIS messages we have...")
print("Total number of AIS messages: ", aisdk.df.shape[0])
```

```
After keeping only 'Cargo', 'Tanker' or 'Passenger' AIS messages we have...
Total number of AIS messages:  191197
```

Figure 3: Pandas DataFrame operations on the MobiML dataset object to keep only cargo, tanker, and passenger vessel types.

```
aisdk = TrajectorySplitter(aisdk).split(observation_gap=timedelta(hours=2))
print("After splitting trajectories with observation gaps we have...")
print("Total number of AIS messages: ", aisdk.df.shape[0])
```

```
After splitting trajectories with observation gaps we have...
Total number of AIS messages:  188118
```

Figure 4: Split trajectories with observation gaps over two hours.

```
aisdk = TrajectoryFilter(aisdk).filter_min_pts(min_pts=20)
print("After removing trajectories with too few points we have...")
print("Total number of AIS messages: ", aisdk.df.shape[0])
```

```
After removing trajectories with too few points we have...
Total number of AIS messages:  188037
```

Figure 5: Drop trajectories with fewer than 20 points.

```
aisdk = TrajectoryDownsampler(aisdk).subsample(min_dt_sec=60)
print("After subsampling AIS tracks we have...")
print("Total number of AIS messages: ", aisdk.df.shape[0])
```

```
100%|██████████| 185/185 [00:00<00:00, 837.62it/s]
After subsampling AIS tracks we have...
Total number of AIS messages:  20623
```

Figure 6: Subsample AIS tracks to reduce the reporting interval between messages to 60 seconds.

4.3. Step 3: Training

To prepare the training stage, we split the dataset using the `TemporalSplitter` (Figure 7). In this example, we use the default: 70/20/10 split. The split subsets are then transformed into a format suitable for training the model. This training itself is controlled by a dedicated configuration, as shown in Figure 8.

```
aisdk = TemporalSplitter(aisdk).split_hr()
aisdk.df
```

```
2024-10-22 22:11:39.705099 Splitting dataset by hours ...
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
train: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],
Train @(min(train_hr), max(train_hr))=(0, 15),
Dev @(min(dev_hr), max(dev_hr))=(16, 20),
Test @(min(test_hr), max(test_hr))=(21, 23)
```

Figure 7: Temporal train/valid/test split.

Once the model is trained, the training result can be visualized on a map, as shown in Figure 9.

```
import mobiml.models.geotracknet.runners as runners
from mobiml.models.geotracknet.flags_config import config

print(config.trainingset_path)
fh = logging.FileHandler(os.path.join(config.logdir, config.log_filename + ".log"))
tf.logging.set_verbosity(tf.logging.INFO)
# get TF logger
logger = logging.getLogger("tensorflow")
logger.addHandler(fh)
runners.run_train(config)
```

Figure 8: Training the GeoTrackNet embedding layer.

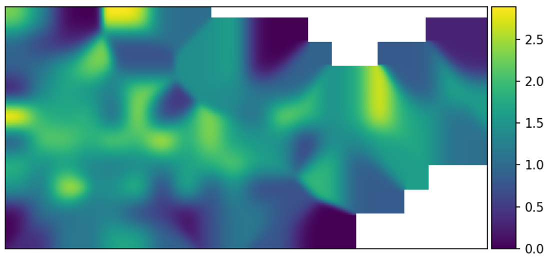

Figure 9: Trained GeoTrackNet logprob map.

4.4. Step 4: Inference

Finally, using the trained model, we can perform the inference by running the GeoTrackNet contrario detection step which flags anomalous trajectories such as the example shown in red in Figure 10.

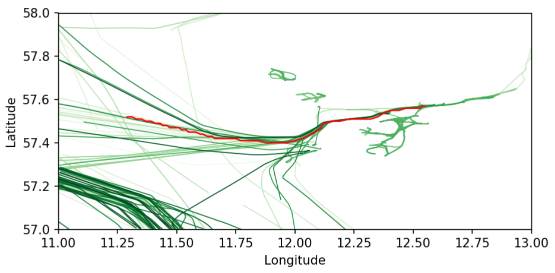

Figure 10: Anomalous track (red) as identified by the GeoTrackNet contrario detection step.

5. Conclusions & Outlook

In this paper, we introduced MobiML, a new library designed to facilitate the development of machine learning (ML) solutions in the Mobility Data Science domain. Our demonstration of the GeoTrackNet anomaly detection model highlights MobiML's ability to support reproducibility and efficiency in mobility-focused ML research. By providing modular components for preprocessing, transformation, sampling, and modeling, MobiML simplifies the development of ML workflows, reducing the effort required to implement key steps. This should also help improve the reliability of research code, since we have ensured that MobiML is covered by unit tests, reinforcing its usability for the broader research community.

Moving forward, future developments should focus on expanding functionality and improving interoperability. Enhancements may include new dataset types, such as flow-based mobility data, as well as improvements in tool harmonization and packaging via PyPI and Conda-Forge. Additionally, we invite ML developers in the Mobility Data Science community to engage with us, contribute to MobiML, and provide feedback to refine its architecture. One of the major challenges in developing MobiML has been managing incompatible Python environments, given the specific requirements of various ML models and libraries. This issue, which affects reproducibility across the ML field, calls for collaborative efforts to find sustainable solutions. Addressing this challenge will be essential to ensuring scalability and long-term usability of MobiML in diverse research and application settings.

Acknowledgments

This work is mainly funded by the EU's Horizon Europe research and innovation program under Grant No. 101070279 MobiSpaces.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] A. Graser, A. Jalali, J. Lampert, A. Weißenfeld, K. Janowicz, MobilityDL: a review of deep learning from trajectory data, GeoInformatica (2024). URL: https://doi.org/10.1007/s10707-024-00518-8. doi:10.1007/s10707-024-00518-8.

[2] M. Mokbel, M. Sakr, L. Xiong, A. Züfle, J. Almeida, T. Anderson, W. Aref, G. Andrienko, N. Andrienko, Y. Cao, S. Chawla, R. Cheng, P. Chrysanthis, X. Fei, G. Ghinita, A. Graser, D. Gunopulos, C. Jensen, J.-S. Kim, K.-S. Kim, P. Kröger, J. Krumm, J. Lauer, A. Magdy, M. Nascimento, S. Ravada, M. Renz, D. Sacharidis, C. Shahabi, F. Salim, M. Sarwat, M. Schoemans, B. Speckmann, E. Tanin, Y. Theodoridis, K. Torp, G. Trajcevski, M. van Kreveld, C. Wenk, M. Werner, R. Wong, S. Wu, J. Xu, M. Youssef, D. Zeinalipour, M. Zhang, E. Zimányi, Mobility Data Science (Dagstuhl Seminar 22021), Dagstuhl Reports 12 (2022) 1–34. URL: https://drops.dagstuhl.de/entities/document/10.4230/DagRep.12.1.1. doi:10.4230/DagRep.12.1.1.

[3] K. Chowdhury, M. Sarwat, GeoTorch: A spatiotemporal deep learning framework, SIGSPATIAL '22, Association for Computing Machinery, 2022. URL: https://doi.org/10.1145/3557915.3561036. doi:10.1145/3557915.3561036.

[4] A. J. Stewart, C. Robinson, I. A. Corley, A. Ortiz, J. M. Lavista Ferres, A. Banerjee, TorchGeo: Deep learning with geospatial data, in: Proceedings of the 30th International Conference on Advances in Geographic Information Systems, SIGSPATIAL '22, Association for Computing Machinery, Seattle, Washington, 2022, pp. 1–12. URL: https://dl.acm.org/doi/10.1145/3557915.3560953. doi:10.1145/3557915.3560953.

[5] A. Graser, MovingPandas: Efficient Structures for Movement Data in Python, GI_Forum – Journal of Geographic Information Science 7 (2019) 54–68. URL: https://hw.oeaw.ac.at?arp=0x003aba2b. doi:10.1553/giscience2019_01_s54.

[6] A. Graser, R. Bell, I. Ilyankou, G. Merten, G. P. Boiko, A. Parnell, L. Vladimirov, Marina, A. Pulver, R. Lovelace, Naarlack, M. Šedivý, J. L. C. Rodríguez, J. B. Elcinto, G. Richter, G. S. Theodoropoulos, B. A. MacGabhann, tsuga, S. Marin, rlukevie, radical squared, git it, Susan, S. Menegon, S. Jozefowicz, R. Miguel, R. Farid, M. Kuhn, M. Fleischmann, L. Delucchi, movingpandas/movingpandas: v0.21.2, 2025. URL: https://doi.org/10.5281/zenodo.14618540. doi:10.5281/zenodo.14618540.

[7] J. V. den Bossche, K. Jordahl, M. Fleischmann, M. Richards, J. McBride, J. Wasserman, A. G. Badaracco, A. D. Snow, B. Ward, J. Tratner, J. Gerard, M. Perry, cjqf, G. A. Hjelle, M. Taves, E. ter Hoeven, M. Cochran, R. Bell, rraymondgh, M. Bartos, P. Roggemans, L. Culbertson, G. Caria, N. Y. Tan, N. Eubank, sangarshanan, J. Flavin, S. Rey, J. Gardiner, geopandas/-geopandas: v1.0.1, 2024. URL: https://doi.org/10.5281/zenodo.12625316. doi:10.5281/zenodo.12625316.

[8] PyMEOS contributors, PyMEOS documentation, 2025. URL: https://pymeos.readthedocs.io/en/latest/.

[9] E. Zimányi, M. M. Duarte, V. Diví, MEOS: An open source library for mobility data management., in: EDBT, 2024, pp. 810–813.

[10] A. Tritsarolis, N. Pelekis, K. Bereta, D. Zissis, Y. Theodoridis, On vessel location forecasting and the effect of federated learning, in: Proceedings of the 25th Conference on Mobile Data Management (MDM), 2024.

[11] D. Nguyen, R. Vadaine, G. Hajduch, R. Garello, R. Fablet, GeoTrackNet—a maritime anomaly detector using probabilistic neural network representation of ais tracks and a contrario detection, IEEE Transactions on Intelligent Transportation Systems 23 (2021) 5655–5667.