A Dashboard Tool for Mobility Data Mining Preprocessing Tasks

Yaksh J. Haranwala
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada
yjharanwala@mun.ca

Salman Haidri
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada
shaidri@mun.ca

Terrence S. Tricco
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada
tstricco@mun.ca

Vinicius P. da Fonseca
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada
vpradodafons@mun.ca

Amilcar Soares
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada
amilcarsj@mun.ca

***Abstract*—Mobility data mining has received significant interest in the literature in the last few years since social media, sensor networks, IoT, and GPS devices generate a vast amount of data. Its growth was also boosted by the growing availability of machine learning algorithms and Python libraries for trajectory analysis. However, we believe that a proper tool that supports trajectory data preprocessing tasks using a dashboard-like application is missing. Such a tool helps users visualize the effects of preprocessing techniques and adequately select the ones that have a desired effect on the data. This demo proposes a tool that combines state-of-the-art Python trajectory analysis libraries to preprocess trajectory data and visualize their effect using a dashboard with maps, tables, and charts that will assist the user through this challenging process.**

***Index Terms*—Mobility data mining, trajectory preprocessing, feature extraction and selection, trajectory visualization**

I. Introduction

Movement data represent trajectories of moving objects. These trajectories are characterized by sequences of spatial positions recorded over time. Analyzing movement data and extracting valuable insights is a challenging task [1]. Movement data is often stored in a complex format and is produced in significant volumes. Additionally, the technology used to collect movement data (e.g., GPS, GSM, Wifi, RFID) often contains errors arising from, for example, device failure or connectivity problems. Nowadays, performing mobility data mining includes a sequence of steps that different computational libraries may handle. This variety of methods makes the data representation and feature extraction a nonhomogeneous process with different inputs and outputs, making the whole data analysis process cumbersome. In this demo, we present a Graphical User Interface (GUI) tool that ties together libraries for handling and exploring preprocessing tasks of trajectory data.

II. Tool Overview

In this section, we present the system architecture of our tool and a description of the functionalities provided in order to showcase what is offered to scientists and researchers.

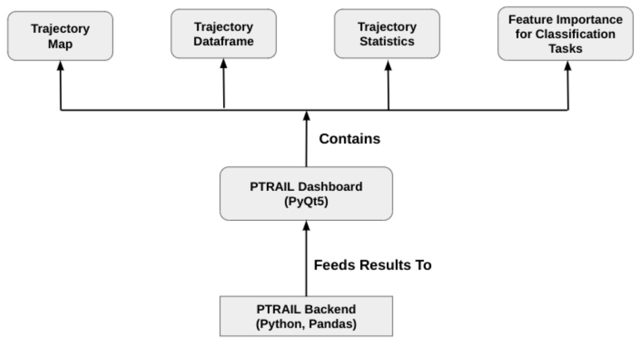
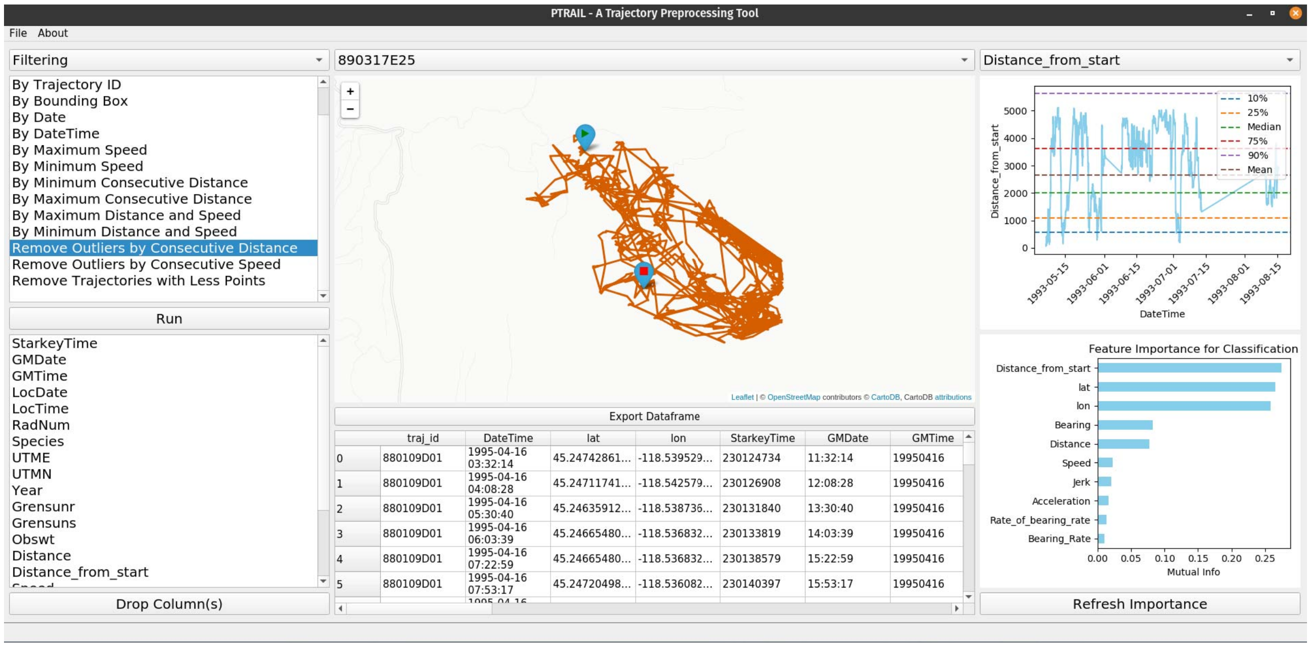


Fig. 1: System Architechture

A. System Architecture

The tool presented in this demonstration was developed using the PyQt5 open-source Python framework for building GUI applications. The core functionalities used in the backend of this tool uses PTRAIL [2], a parallelized Python package for preprocessing trajectory data. As seen in Fig. 1, the tool dashboard consists of four individual components (i.e., views), namely: (i) Trajectory Map, (ii) Trajectory Dataframe, (iii) Trajectory Statistics, and (iv) Feature Importance visualizer for classification tasks. The map is used to visualize the trajectory of one object at a time from the dataset, and it can be utilized to visualize the paths of all other objects present in the dataset by clicking on the Trajectory Dataframe viewer. The Trajectory Dataframe component is used to organize and visualize the raw data present in the dataset and any features generated by the tool. The tool also has Trajectory Statistics, which is a line chart that compares the kinematic features of one moving object to others. It's goal is to visualize and decide which methods can be used to perform data cleaning. Finally, if the



	traj_id	DateTime	lat	lon	StarkeyTime	GMDate	GMTime
0	880109D01	1995-04-16 03:32:14	45.24742861...	-118.539529...	230124734	11:32:14	19950416
1	880109D01	1995-04-16 04:08:28	45.24711741...	-118.542579...	230126908	12:08:28	19950416
2	880109D01	1995-04-16 05:30:40	45.24635912...	-118.538736...	230131840	13:30:40	19950416
3	880109D01	1995-04-16 06:03:39	45.24665480...	-118.536832...	230133819	14:03:39	19950416
4	880109D01	1995-04-16 07:22:59	45.24665480...	-118.536832...	230138579	15:22:59	19950416
5	880109D01	1995-04-16 07:53:17	45.24720498...	-118.536082...	230140397	15:53:17	19950416

Fig. 2: The Dashboard View

loaded dataset is suitable for classification tasks, the user can generate a features' mutual importance score and filter out irrelevant features.

B. Functionalities

Data can be loaded into the dashboard easily from CSV files and various tasks can be performed on the data in order to remove noise and errors from it. The functionalities provided by the tool are described below.

1) Feature Extraction: In order to better utilize and enrich trajectory data, we provide several temporal features, such as date, time, year, or day of the week, as well as kinematic features, such as speed, acceleration, or bearing, that can be extracted from the data to exploit it for machine learning tasks such as classification and regression. By creating these new features, we can better summarize the information contained in the raw movement data. The GUI provides a simple method to extract those features. As can be seen in Fig. 2, the user can utilize the drop-down menu to choose Kinematic or Temporal features and either select the option to produce all of them at once or an individual function. The feature(s) are generated in parallel, and a message is displayed in the bottom left corner once the task is complete. The Dataframe is updated with these new features and is open to further analysis by the user. Once the features are generated, the user can compare the features of one object versus all the others in the dataset using the Statistics panel on the left side of the dashboard, as seen in Fig. 2. After, the user may decide whether to apply further cleaning or interpolation methods to smooth the data to obtain the desired results.

2) Trajectory Filtering: Since trajectory data is collected from various sources like GPS or animal collars, it is prone to contain errors caused by inaccurate location, faulty sensors, or unstable connections. Hence, this tool offers several filtering strategies such as Hampel filter and filtering based on features like speed or acceleration to clean such error-ridden data. In order to extract a fragment of the Dataframe, other filtering methods such as filtration by date, speed, time, and distance are also provided. Our tool employs outlier detection based on distance and speed to handle extreme values and uses Hampel filter. All of these filtering options are available to the user from the drop-down menu in the top left corner of the GUI and pressing run. After, the user is prompted to enter the threshold parameters according to the filtering option they choose. Once they press okay, the task is executed and the datframe and the folium plot of the trajectory is updated for the user to exploit.

3) Trajectory Interpolation: PTRAIL further extends the pipeline of trajectory data cleansing by providing parallel versions of scipy's [4] cubic and linear interpolation techniques along with random-walk [6] and kinematic [5] interpolation. Using the functionality selection drop-down menu in the GUI, the user can select any of the four interpolation methods. By pressing run, the user is prompted to enter the sampling rate, and on pressing okay, the task of interpolation begins. The interpolated points are visualized, the folium plot is updated along with the Dataframe, and the user can see all the interpolated points on the Dataframe.

4) Statistics Generation and Feature Ranking: In several situations, continuous trajectory data is split into smaller parts to examine the local behaviors in isolation and gain a deeper understanding of the movement pattern of the object in question. This process is called trajectory segmentation, and basic methods to achieve it are provided in the tool.

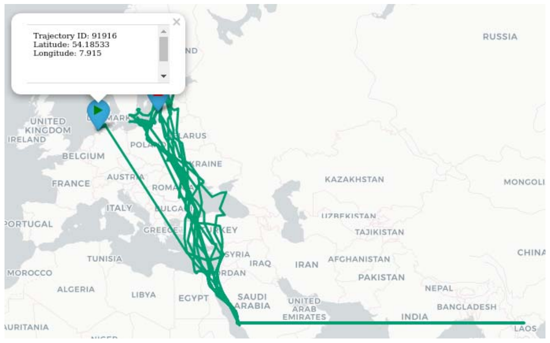


Fig. 3: Original Trajectory

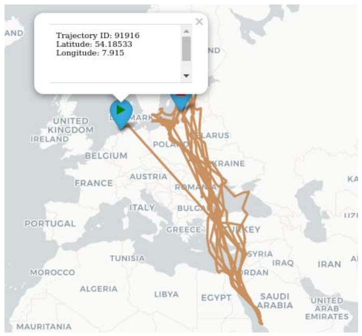


Fig. 4: Trajectory with anomalies removed

The tool provides functionality to split the data into smaller segments based on time. Furthermore, recent trends in the industry have focused highly on machine learning tasks such as trajectory classification of objects on movement data. Our tool also integrates the scikit-learn's mutual information feature that calculates how much mutual information each variable generated by the tool has. A bar chart is displayed ranking the features in descending order to show the importance of each one. This can help the user decide only to keep the features relevant for a classification task.

5) Data Transformation and Export: Generally, movement data is stored using a point-based representation. The sequential observations of an object over several instants of time and other features are the values recorded using a point-based representation. It is to be noted that the data stored in this format is often deemed unsuitable for machine learning tasks as it cannot characterize the spatio-temporal dependencies of the object's movement, which are relevant to several applications like transportation mean and fishing activities detection. Therefore, many problems need a swap from a point-based to a segment-based representation. In a segment-based data representation, instead of characterizing the behavior of the movement in a single trajectory point, we group the trajectory points using some partitioning criteria. In our tool, we provide the functionality to extract features from the trajectory points and transform the data into a segment-based representation with the click of a button. Finally, our tool provides the functionality of writing the current state of the Dataframe to a CSV file. Therefore the outcome can be used by several other tools able to handle this format.

III. Demonstration

In this demonstration, we use the Starkey [7] dataset, which contains the movement data of elk, deer, and cattle for a period from 1993 to 1996. The Starkey dataset is provided as one of the base datasets on PTRAIL's github repository and other datasets. PTRAIL has examples with several other datasets, such as the Atlantic Hurricane Dataset [8] and the Bird Tracking dataset [9], that can also be used for our tool demonstration.

The Starkey [7] dataset is loaded onto the system, and Fig. 2 shows the trajectory of a single animal in the center of the dashboard (i.e., Trajectory map) along with the Trajectory DataFrame viewer below it. The user can explore all the trajectories available in the dataset by clicking on the table rows at the tool's bottom. The main goal is to make it easier to compare different trajectories present in the loaded dataset. Furthermore, the maps are generated with folium and are interactive to visualize additional information such as the start and endpoints of the trajectory.

Initially, when the data is loaded into the tool, it is filled with outliers as can be seen in Fig.3. By inspecting it, we can clearly see some outliers. Another way to inspect anomalies is through the inspection of features values. When kinematic features are generated on the dataset, the anomalies can be visualized on the top right corner of the tool, where the trend of specific metrics compared to other objects in the dataset is shown. The option to compare each feature is added automatically when the feature is generated using the tool. This helps the user further visualize the level of noise and anomalies in the dataset and decide to use other methods to clean the data.

Fig.5 shows the speed feature for a single animal and the comparison of that particular animal to the others present in the dataset. Fig.5 also indicates the presence of noise and several irregularities in the dataset. On the other hand, Fig.6 displays the same trajectory after removing outliers based on speed and running the Hampel filter on the dataset premised on acceleration and speed, and finally smoothening the dataset using cubic interpolation, one of the four interpolation methods available. The statistics after the anomaly removal and trajectory smoothening can be clearly seen to display a more regular pattern of speeds on consecutive days. The use of outlier detection and removal is further shown by Fig.4 wherein it can be clearly seen that there are no more sudden jumps in the animal's movement as compared to Fig.3 which shows the plot before running the filters.

Finally, after completing the data filtration and interpolation of the Starkey dataset, we can select features for a classification task using the species of the animals as the target feature. Therefore, to decide which features to retain before exporting the dataset, we generated the mutual information bar chart as shown in Fig.7 and Fig.8. Fig.7 represents point-based representation Mutual information and Fig.8 represents

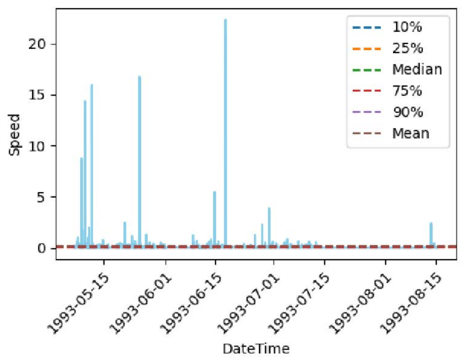


Fig. 5: Speed of a single animal vs. others in the dataset before preprocessing

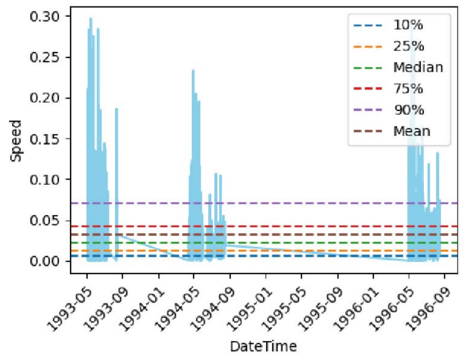


Fig. 6: Speed of a single animal vs. others in the dataset after preprocessing

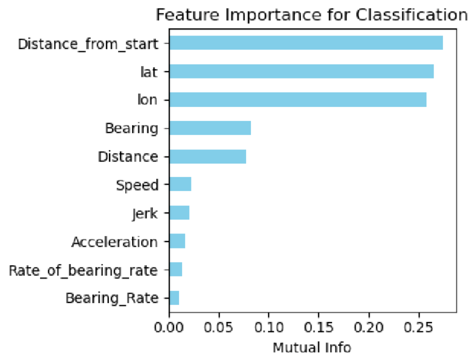


Fig. 7: Point-based Mutual Info

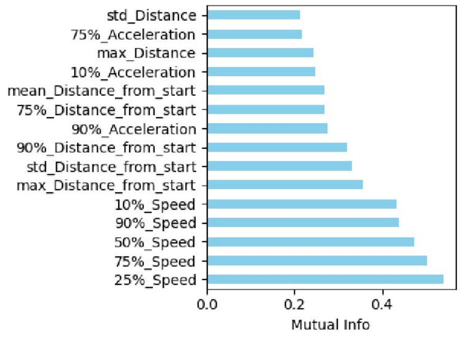


Fig. 8: Segment-based Mutual Info

segment-based representation Mutual information between the features generated by the tool. The figures clearly show that the segment-based view of the trajectory data is more likely to yield better results for classifying the animals' species.

IV. Conclusion

In this paper, we demonstrated a GUI tool that is used for trajectory data preprocessing tasks. The GUI tool offers several salient data preprocessing mechanisms such as filtering, feature extraction, trajectory visualization using maps, statistical comparisons, and feature importance for mobility data mining tasks. Our proposed tool aims to cut down on time and efforts spent on preprocessing of movement data and adds several visualization components that aim to make the analysis of the data easier and assist in the decision of selecting features for machine learning tasks.

References

[1] L. M. Petry, et al. Challenges in vessel behavior and anomaly detection: From classical machine learning to deep learning. Canadian Conference on Artificial Intelligence. Springer, Cham, 2020.

[2] Salman Haidri, Yaksh J Haranwala, Vania Bogorny, Chiara Renso, Vinicius Prado da Fonseca, and Amilcar Soares. 2021. PTRAIL–A python package for parallel trajectory data preprocessing. arXiv preprint arXiv:2108.13202 (2021).

[3] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[4] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

[5] J. A. Long, Kinematic interpolation of movement data, International Journal of Geographical Information Science 30 (5) (2016) 854–868.

[6] G. Technitis, W. Othman, K. Safi, R. Weibel, From a to b, randomly: a point-to-point random trajectory generator for animal movement, International Journal of Geographical Information Science 29 (6) (2015) 912–934.

[7] Wisdom, Michael J., technical editor. 2005. The Starkey Project: a synthesis of long-term studies of elk and mule deer. Lawrence, Kansas: Alliance Communications Group. 252 p.

[8] Landsea, C. W. and J. L. Franklin, 2013: Atlantic Hurricane Database Uncertainty and Presentation of a New Database Format. Mon. Wea. Rev., 141, 3576-3592.

[9] Stienen E W, Desmet P, Aelterman B, Courtens W, Feys S, Vanermen N, Verstraete H, Van de walle M, Deneudt K, Hernandez F, Houthoofdt R, Vanhoorne B, Bouten W, Buijs R, Kavelaars M M, Müller W, Herman D, Matheve H, Sotillo A, Lens L (2017). Bird tracking - GPS tracking of Lesser Black-backed Gulls and Herring Gulls breeding at the southern North Sea coast. Version 5.6. Research Institute for Nature and Forest (INBO). Occurrence dataset https://doi.org/10.15468/02omly accessed via GBIF.org on 2022-02-24.