ELEN0062 - Introduction to Machine Learning
Project 3 - Competition

November 20th, 2023

The aim of this project is to get some experience with designing a solution to a more complex problem on large real data, using the different tools seen in the course. You will be able to compare the performance of your solution with other groups on a leaderboard during this competition, although it is obviously not the sole grading factor.

The code and data can be found on the projects website: `https://iml.isach.be`. Once again, the submissions will be done on Gradescope[1]. The project must be carried out by groups of at most three students. See below for the deadlines.

1 Introduction

Over time, taxi technologies have evolved, notably the use of electronic dispatch systems. With these systems, taxis provide GPS information about their location, but usually do not provide their destination. Therefore, it might be hard to know which taxi to contact when planning a future trip. We would like to tackle this issue, so that we can know which taxis will end near a new requested pick up. This would highly improve the efficiency of taxi systems.

More precisely, the goal here is to design a solution that **predicts the destination of a taxi trip given its partial initial trajectory**, which is of varying length. For that, you have access to a database of full trajectories, which you can use to train your solution.

Along with the trajectories which can certainly help predicting the destination, you have access to some metadata, such as the taxi ID, the phone ID of the customer when the taxi was ordered with a call, and more. These pieces of information could also perhaps help in narrowing down the destination based on the historical training data.

For the predictions, the information the model has access to is in the same format, except the trajectory is now only a cut-off initial trajectory of varying length.

2 Data

The dataset comprises a full year of trajectories for all the 442 taxis in Porto, Portugal. This results in a table of 1,710,670 rows.

The taxis operate through a taxi dispatch central, using mobile data terminals installed in the vehicles. Each ride is categorized into 3 categories:

A) Taxi central-based: Trip dispatched by the central. An anonymized ID is provided, when such information is available from the phone call.

B) Stand-based: Trip directly demanded on a taxi stand.

C) Non-taxi central-based: Trip directly demanded on a random street. The data, which you can find in the `train.csv` file, consists of a table with nine features, which are summarized in Table 1.

[1] `https://www.gradescope.com`, Entry code: JK485B.

Feature	Type	Values	Description
TRIP_ID	String		Identifier for each trip. *(Some are not unique!)*
CALL_TYPE	Character	A, B, or C	Category of the trip, as detailed above
ORIGIN_CALL	Integer	Can be NaN	Unique identifier associated to a phone number that was used to order a taxi trip at least once. Only included if CALL_TYPE is A, otherwise is NaN.
ORIGIN_STAND	Integer	Can be NaN	Unique identifier for the taxi stand, only included when CALL_TYPE is B. Otherwise, contains NaN.
TAXI_ID	Integer		Unique identifier associated to the taxi driver of the trip.
TIMESTAMP	Integer		Unix timestamp of the trip (in seconds). Identifies the start of the trip.
DAY_TYPE	Character	A, B, or C	Identifies the type of the day for the trip. · A: Normal days. · B: Holidays. · C: Days before B-type days.
MISSING_DATA	Boolean	True/False	Contains False if the GPS stream (in POLYLINE) is complete, and True if there are missing data points in it. Note that it is part of the challenge to deal with this knowledge.
POLYLINE	List as a string		Contains a list of GPS coordinates (i.e. WGS84 format) mapped as a string. The beginning and the end of the string are identified with brackets (i.e. [and] respectively). Each pair of coordinates is also identified by the same brackets as [LONGITUDE, LATITUDE]. This list contains one pair of coordinates for each 15 seconds of trip. The last item corresponds to the trip's destination, while the first one represents its start.

Table 1: Data table format for the training data (train.csv). The test data (test.csv) follows the same format, except the POLYLINE data is only the (partial) initial trajectory.

3 Target Prediction & Test Data

Given the partial initial trajectory, *which is of varying length*, you have to predict the destination. It is represented by two values, the LATITUDE and LONGITUDE, in the same WGS84 format as the training data, i.e. degrees.

You are given a test set (in test.csv) with the same data format as the training data, except the trajectories are therefore now *cut-off*. It contains 320 rows with partial initial trajectories. Note that the test data is based on trips that have happened in the year after the one recorded in the training data.

Your task is to compute the destination for each of the 320 rows, i.e. the final latitude and longitude. Your values should be in the format:

```
TRIP_ID,LATITUDE,LONGITUDE
"T1",41.146504,-8.611317
...
"T24",41.146504,-8.611317
...
```

The data types are therefore: String, float, float. Pay attention to the differences with the POLYLINE in the CSV tables, for the order of the values. These output should be saved in a .csv file, similar to the training data. An example submission file (example_submission.csv) is included, along with a function (in example_script.py) to easily produce your submission file.

4 Submission

The goal of the competition is to reach the best possible score (i.e., lowest mean Haversine distance, see below) on the test set provided in `test.csv`.

Your predictions computed on the data in `test.csv` can be submitted multiple times on Gradescope, where a leaderboard will allow you to compare your methods to other groups. A function to write your results in a CSV file is provided in `example_script.py`.

When submitting *during the competition*, public scores will be computed on about 50% of the total test set. We will obviously not disclose which subset this is. *Once the competition is over*, new private scores will be computed on the other 50% of the set. The private scores will count as the final scores.

5 Evaluation Metric

Predicted destinations are compared to the ground truth using the commonly-used Haversine distance. It compares two points on a sphere based on their latitude and longitude. The distance is computed as:

$$d = 2 \times 6371 \times \arctan\left(\sqrt{\frac{a}{1-a}}\right)$$

where a is computed by:

$$a = \sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\left(\phi_1\right) \times \cos\left(\phi_2\right) \times \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$$

where:

- ϕ_i: Latitude for point i (in radians!).
- λ_i: Longitude for point i (in radians!).
- 6371 is the radius of Earth (in km).

The distance is computed between each pair of prediction (ϕ_1, λ_1) and true value (ϕ_2, λ_2). Then, we take the mean distance over all predictions. You can therefore interpret this score as the average error in kilometers. A function that computes it is provided in `metrics.py`.

6 Report

By ***December 13th***, you should have submitted a report along with your code on Gradescope.

Your report should describe your investigations on the data and on different approaches, along with your final approach and results. It *must* contain the following:

- A detailed description of all the approaches you have investigated.
- A detailed description of your approach to select and assess your model.
- A table summarising the results of your different approaches.
- All tables, figures and results should be analysed in-depth while avoiding unnecessary redundancies.
- Any complementary information that you want to analyse.

Please note that the methodology and the quality of your report matter a lot. Great results on the leaderboard with poor motivations, explanations, and details in the report are definitely not what you should aim for.

7 Rules

You have to *strictly adhere* to the following rules:

- You can use any techniques or algorithms you want.
- Solutions can be implemented in any language using any libraries. The only condition is that your solution should be strictly reproducible.
- You can not use external data, unless you get our approval first.
- If you use external code for algorithms, you have to give the references (even for tools or algorithms used or developed in previous projects such as Scikit-Learn).
- You can not use ready-made solutions for the problem, i.e. any available software specifically designed to solve the problem of taxi destination prediction.