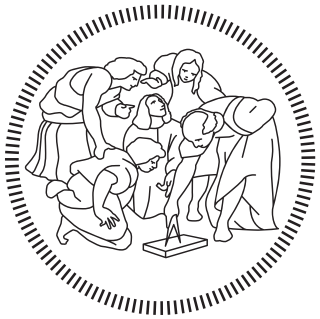
POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Master of Science in Mobility Engineering



Machine Learning applied to raw gps trajectories to enable transport mode detection

Academic supervisor: Prof. Mark James Carman

Gabriele Rossi, 953431

Academic year: 2022-2023

Abstract

The number of active devices in our pocket has exploded since 2008, with it, also the magnitude of data coming from smartphone sensors. A project started by Microsoft in 2005 collected gps data from people moving around different cities, in particular Beijing. That project has had a crucial role in enabling the research in transport mode detection algorithms starting from raw gps data. Nowadays gps is not the only positioning system available, phone companies are beginning to sell low resolution trajectories of their costumers, and other technologies are providing this type of data at very low cost.
In this framework, there is a big opportunity to get rid of traditional methods to gather informations about commuters habits. These methods, such as censuses, questionnaires… are resource consuming and could be substituted or complemented by more efficient methods.
The added value of such informations can range from support to urban planning, to efficient scheduling of activities, services, transports, etc…
To make the most of the potential of this system, it is useful to create a transport mode detection algorithm, capable of doing data mining from the raw data and obtaining as much information as possible from it. This work highlights the limits due to the scarcity of a ground truth on which calibrate the algorithms but also the accuracy that can be reached when the data is abundant.

Abstract

Il numero di dispositivi elettronici attivi nelle nostre tasche è esploso dal 2008, con esso anche l'entità dei dati provenienti dai sensori degli smartphone. Un progetto avviato da Microsoft nel 2005 ha raccolto dati GPS da persone che si spostano in diverse città, in particolare Pechino. Quel progetto ha avuto un ruolo cruciale nel consentire la ricerca di algoritmi di rilevamento della modalità di trasporto a partire da dati GPS grezzi. Al giorno d'oggi il gps non è l'unico sistema di localizzazione disponibile, le compagnie telefoniche stanno iniziando a vendere traiettorie a bassa risoluzione dei loro clienti e altre tecnologie stanno fornendo questo tipo di dati a costi molto bassi.
In questo quadro, c'è una grande opportunità per accantonare dei metodi tradizionali per raccogliere informazioni sulle abitudini dei pendolari. Questi metodi, come censimenti, questionari... consumano risorse e potrebbero essere sostituiti o integrati con metodi più efficienti.
Il valore aggiunto di tali informazioni può spaziare dal supporto alla pianificazione urbanistica, alla programmazione efficiente di attività, servizi, trasporti, ecc...
Per sfruttare al meglio le potenzialità di questo sistema, è utile creare un algoritmo di rilevamento della modalità di trasporto, in grado di fare data mining dai dati grezzi e ricavarne quante più informazioni possibili. Questo lavoro mette in evidenza i limiti dovuti alla scarsità di una verità di base su cui calibrare l'algoritmo ma anche l'accuratezza che si può raggiungere quando i dati sono abbondanti.

List of Figures

List of tables

1. Introduction

As long as humanity will feel the need to move people and goods, there will be a transport demand and a diversified supply of transport modes able to met that demand.
Mobility, intended as the easiness to move people and goods, is a backbone for our society because is rooted in many aspect of our everyday life, it is able to shape the development of a community both from a social and economical point of view.
We can identify two milestones regarding the research in this field. The first is the Wardrop research about user equilibrium[1] that brought to us a strong model to describe the spreading of trips over alternate routes because of congested conditions. Since then, transport and urban planning has been seen as an isolated task, the demand was considered inelastic and the goal was to maximize the efficiency, planning the right services and infrastructures to connect origins and destinations.
Getting close to the 21th century, LUTI models [2] (land use and transportation interactions) emerged, whose goal was to introduce the real world complexity: transport systems are not isolated, they influence and are influenced by neighboring systems such as housing and labour markets, in other words, the supply influence the demand and viceversa.
Since that dates mobility studies exploded, researchers, companies and governments recognize in this field the power not only to push for economic growth, but also to study human behavior in general, epidemic modeling, internal security [3] and income segregation [4] to mention some.
Thanks also to modern technologies and tools, different papers are demonstrating the weaknesses and strengths of traditional models but also proposing updates to make them more reliable. For example it's been discovered that people don't follow the routing behavior described by Wardrop [1], they don't minimize the path cost but are affected by some bias and interferences. Previous literature has tried to identify causes behind deviation from cost minimization, finding that factors that influence this are several and related to many aspects, for example, the initial straightness of the route, the relative topography, the presence of landmarks and anchor points, the direction and other aspects influenced by estimation errors.[5]
Another intuition enabled by high resolution location data is how segregation spreads and holds inside our cities, if once it was well described by the neighborhood of residence, now is important to extract fine grained mobility patterns to distinguish different income groups, because different groups live often close to each other but visit different places for different reasons.[4]

2. Scope

In this framework, the goal of my work is to make easier data mining from gps trajectories, in particular we need a method to count real vehicular and human flows without exploiting surveys and census data. These last methods are the traditional ones and for their being resource-intensive need to be replaced or complemented by newer ways of data collection. In this thesis we will discover that transport modes can be recognized exploiting machine learning algorithm applied to gps trajectories. These are collected through mobile devices that are pervasive and ideally provide very fine grained information in time and space, then I will measure how data collected through these devices is representative of population accounted by public institutions (i.e. ISTAT's OD matrices [6]).

My work developed following this chronological roadmap:

- I reproduced the experiment of Microsoft geolife project [7] building a classifier able to detect the correct transport mode from a dataset of geospatial trajectories belonging to different users.
- I added some element of novelty in feature engineering and model validation to test the robustness to novel data.
- I generalized the classifier to make it compatible with a novel dataset coming from Cuebiq [8], a company that deals with geospatial data gained from smartphone users.
- Being Cuebiq devoid of a ground truth, a validation method based on istat OD matrices is under construction and will be covered in future developments of this work.

3. Previous works

Inferring the transport mode using georeferenced data is a topic that is linked to the mass adoption of smartphones, but since then, the lack of a ground truth against which validate emerging algorithms its been a problem to which researchers answered in different ways [9]:

- The most common validation method considers mode share statistics and origin-destination matrices: if predicted vehicular flows coincides with the ones coming from official and public sources the model is valid.
- The second most common approach to validation is not to validate but to base the algorithm on a prior knowledge about the dynamics of people and vehicles.
- In the third position we have a ground truth coming from labelled data: for each trajectory there is a declared transport mode.
- Lastly, we find flight data, manual counting and simulated data.

Focusing on labelled data, a vast academic literature dealing with it is available. For this task, different statistical and machine learning methods have been applied [10], for instance ensemble classifiers, hidden Markov chains and deep learning. All these studies declared high level of accuracy: from 75% to 95%, but the results are often favored by the validation criteria and the availability of data from sensors different from gps.
Talking about the first issue there are vehicles which dynamics is intrinsically difficult to disentangle, so, for example, a classifier will likely classify a taxi, a car and a bus as the same vehicle. The result is that merging this three categories into one will return higher metrics.
The second criticality is for that databases that contains informations coming from multiple sensors (gps, accelerometer, gyroscope, rotation vector), also in this case, the performance are higher, but these systems rely on data that are less available.

Up to date, the richest database publicly available of labelled trajectories comes from the Microsoft Geolife experiment [7]. It is the richest in terms of participating users and size of the time window [10]. A set of volunteers kept their gps devices on in order to collect the visited locations, declaring constantly the transport mode chosen to complete their trip. For my work, I trained my classifier on this dataframe.

4. Method

4.1 Dataset

Geolife database is composed both by labelled and unlabelled trajectories, for my work I will analyze only the first type. Positioning and transport mode information lies in two different tipe of files, so, through timestamps and user identification numbers a merge between the two databases is achievable. This will result in a list of point localizable in time and space as shown below (table 1).

time	lat	lon	label	user
2008-03-28 08:44:30	39.962098	116.301595	Bus	104
...	...	...	...	...

Table 1: Original Geolife database, preprocessed to merge the raw gps trajectories with the declared labels.

The samples cover a period of time that goes from 2007 to 2012 with an uneven distribution in time, for a total of over 5 million rows, 64 unique users and 11 different labels: taxi (4.41%); car (9.40%); train (10.19%); subway (5.68%); walk (29.35%); airplane (0.16%); boat (0.06%); bike (17.34%); run (0.03%); motorcycle (0.006%); and bus (23.33%).

As emerged in other works, this labels are not fully trustable because are affected by human error [11]. Therefore, it is possible that some users forget to annotate the trajectory when they switch from one transportation mode to another. Although this is an assumption and it can not be proved, some trajectory samples, some circumstantial evidence, show a continuous behavior change that is suspected to be this type of error.
In the picture below (Figure 1) is clearly visible a transition from speeds that are typical of walking to those that are not compatible with this class, but the entire trip's annotation is "walk".

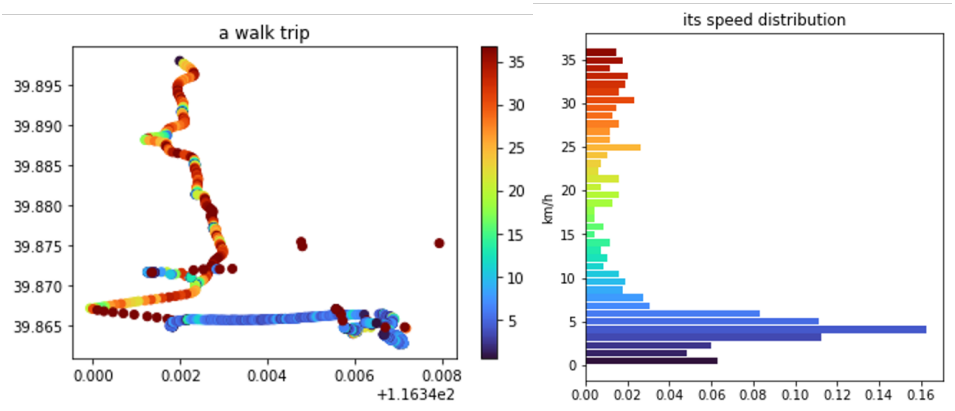


Figure 1: on the left a trajectory labelled as "walk" is represented through longitude and latitude, on the right the relative speed distribution is presented.

More than 80% of datapoints are contained in the area of Beijing (China), for this reason, to have a more homogeneous context to work with, a bbox (bounding box) will be applied to filter outer trips.
The bounding box has the following coordinates: longitude ranges from 116.081035 and 116.751230, while latitude from 39.695257 and 40.212878.
The filtered dataset will contain all modes that start and end their trips inside the above mentioned squared region. An exception to this rule is represented by trains' trips, since they are just 2% of the total ones and their endpoints fall far outside Beijing, all of them will be kept, without filtering.
Lastly, the classes for airplane, boat, run and motorcycle are removed because they contain too few samples to guarantee a meaningful train set.

4.2 Feature engineering

4.2.1 Trip ID

From raw location based data is necessary to extract single trips, I will leverage on the ground truth using the following algorithm:
I sort all the rows according to the user id, and inside each user in a chronological order.
Now the dataframe is a list of subsequences, each one belonging to a different user. Inside those, every time the declared **mode** of transport changes, or if the time occurred between two successive points is major than **one hour**, the subsequence will be further divided into trips. An intuition about this process is illustrated in the figure below (figure 2):

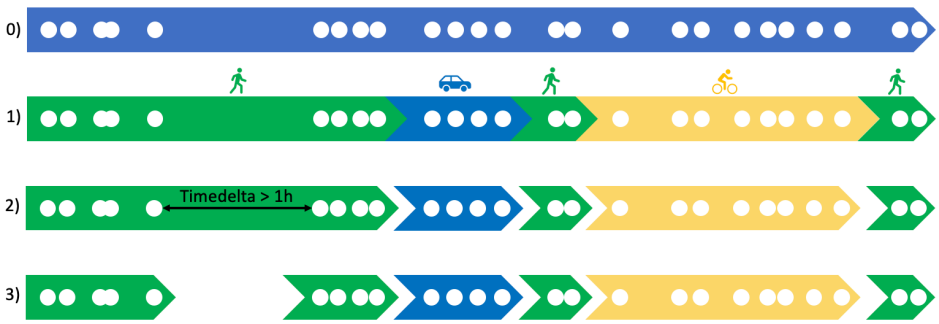


Figure 2: The first arrow 0) is a timeline in which every datapoint belongs to a single user, with the step 1) this sequence is divided according to the different transport modes, with the step 2) these segments are further divided where a time gap major than an hour occur between two points, in the last step 3) a trip identification number is assigned to each segment.

From here, it is important to question the significance of the information we have and take the opportune countermeasures where necessary.
Since we are dealing with transports, we would like to include a great variety of vehicles and contexts, because their behavior is strictly contingent to the location we are considering. For example, regulation imposes different speed limits according to the country. Another limitation to be considered is that trajectories labelled as subways contains points with high density, suggesting that an important fraction of Beijing subways runs at the surface level, consistently with this theory the map below (figure 3) show how related datapoints follow the topography of the city.

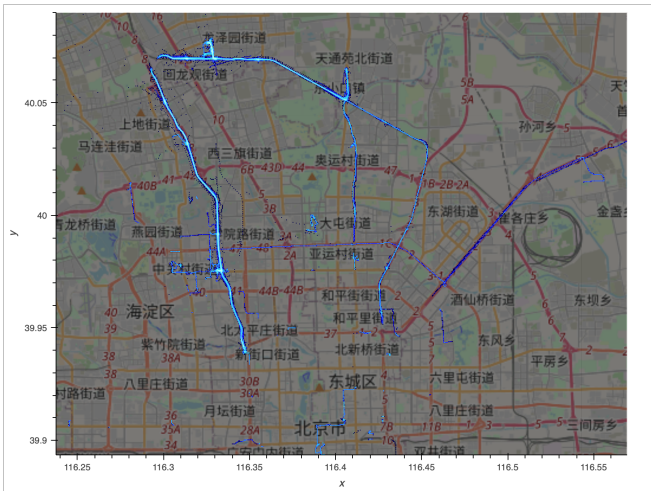


Figure 3: datapoints labelled as “subway”

That, possibly, will translate in poorer performance when a classifier will be applied to Italian cities, where metros runs mainly underground.

Now that points are labelled with their respective trip identification number, it is possible to built the train set that will serve as an input for a **XGboost** classifier [12]. This last machine learning algorithm has proven both in literature and in my experiment to be one of the most powerful tool. The advantage of this algorithm lies also in the fast computation of **SHAP values** [13]. SHAP (SHapley Additive exPlanations) is a method based on cooperative game theory and it is used to increase transparency and interpretability of machine learning models. I will use it to compute and to have a deeper insight on feature importances. Further explanation on this topic will be provided in the *Feature importances* section.

4.2.2 Point-level features

After assigning tripIDs to each point it's now the time to compute physical quantities to describe the variations from the actual point to the next one. Together with variations, local informations such as the closeness to a POI (point of interest) is added in this section. These intermediate quantities are computed for every point and they are:

- **Timedelta** [*s*]: is the time occurred between two subsequent points in a trip. The mode of the distribution is 2 seconds, indeed 98% of the point shows a timedelta minor than 10 seconds. This value gives us idea of the high sampling rate. We have to pay attention to this resolution, because other datasets are more coarse grained, and so a generalization of the algorithm to those data source is not straightforward. Examples of coarse grained datasets concerning human mobility are the ones provided by Cuebiq, Google Maps timeline, and GSM mobile phone data [14].

- **Spacedelta** [*m*]: is the Haversine distance computed between two subsequent points in a trip.

- **Speed** [*km/h*]: is the average speed occurred between two subsequent points, indeed it is computed as:

$$Speed_{i,i+1} = \frac{spacedelta_{i,i+1}}{timedelta_{i,i+1}}$$

  where *i* is the a point of the trajectory and *(i, i+1)* is the segment that goes from point *i* to the next

- **Acceleration** [*m/s^2*]: since we are dealing with a discrete space, instantaneous acceleration cannot be defined, I will use this term to indicate the variation of speed between two subsequent segments divided by the time occurred. Even if it refers to two segments and need three different points to be evaluated, its value will be assigned to the first point (*i*) of the sequence

$$Acceleration_{i,i+2} = \frac{Speed_{i+1,i+2} - Speed_{i,i+1}}{Timedelta_{i,i+1}}$$

  where *i* is the a point of the trajectory, *(i, i+1)* is the first segment and *(i+1,i+2)* is the following one

- **Angle** [° *or arcdegrees*]: is the angle included by the segment that goes from the actual coordinate to the next one and the segment that goes from the previous coordinate to the actual one, computed in a clockwise way. Since trips' endpoints haven't a previous or a subsequent point, an angle cannot be defined, in this case a Nan value will be assigned. Using this method, angle ranges from 0° to 360°, if the path is straight the average will be 180°, if it contains more right turns or left turns, it will tend respectively more towards 0° or 360°. This technical choice makes a distinction between turning left and right. That could possibly increase the predictive power of the model because, as some tasks that require more right turns exist [15], also some mode of transport modes can present a comparable pattern.

The next labels are generated with the help of *overpass-turbo* [16], a web tool that allow to query Open Street Map [17] and download the relevant data for one's research.

In particular, I requested the centroids (in latitude and longitude) of different buildings related to public transit.

- **Train station id**: each point is labelled with the name of a train station if it falls into a circle of radius 300 m from the centroid of that station, if not, it is labelled as “False”.
- **Subway station id**: each point is labelled with the name of a subway station if it falls into a circle of radius 100 m from the centroid of that station, if not, it is labelled as “False”.
- **Bus stop id**: each point is labelled with the name of a stop if it falls into a circle of radius 100 m from the coordinates of that stop, if not, it is labelled as “False”.
- **Airport id**: each point is labelled with the name of an airport if it falls into a circle of radius 500 m from the centroid of that airport building, if not, it is labelled as “False”.

The resulting dataframe has now the following aspect (Table 2):

time	lat	lon	label	user	trip ID	speed	acceleration	angle	tr_st_id	sub_st_id	…
2008-03-28 08:44:30	39.962098	116.301595	3	104	1	23.1343	-0.017825	NaN	FALSE	57	…
2008-03-28 08:48:30	39.948270	116.303298	3	104	1	7.7334	-0.068231	155.0	FALSE	FALSE	…
…	…	…	…	…	…	…	…	…	…	…	…

Table 2: dataframe of processed trip trajectories, the first 5 columns are the original ones, from which the other 6 on the right are derived.

Looking at the empirical distributions of some of this quantities, it starts to be visible that their shape change among different classes. This is particularly true for speed distributions, while cars go at different speeds in an evenly distributed way, some physically constrained vehicles such as trains and subways have peaks around specific values (figure 4).

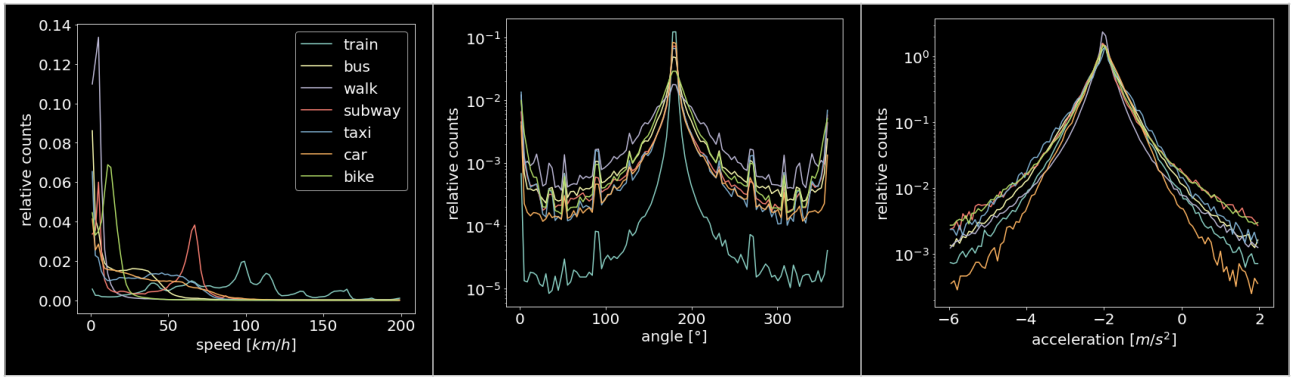


Figure 4: empirical distributions of speed, angle and acceleration.

4.2.3 Trip-level features

The next features comes from intuitions and past researches on this field, their construction starts from the question: “to which quantity I have to look to distinguish a mode of transport from another?”
The goal is to summarize multiple points into one single row describing the trip.
Now, I group the datapoints by tripID, and for each trip I compute the following statistics. All of them can be categorized in groups, according to the physical quantity they are related to.
The first group of trip-level features is related to the cinematic of the user and it’s made up of:

- **Number of points**: is simply the number of samples of which the trip is made of.

- **Trip distance**: is the length, in meters, composing a single trip.
  - **Spacedelta percentiles**: Regarding the distances, other 5 values have been computed to provide the classifier an insight about the distribution that underlies this quantity. Since a trip is made of multiple segments, intended as the vector connecting two subsequent points, we can derive a distribution of spacedelta (as defined in the previous paragraph).
    From this distribution we extract 5 percentiles (**5%**, **25%**, **50%** or the median, **75%**, **95%**).
    Intuitively the predictive power of these features can be explained through few examples: If the 5% percentile of spacedelta is 100 meters, it means that only 5% of segments are shorter than 100 meters, combining this information with the sampling rate that is about 0,5 samples/seconds, we understand that the gps trajectory belong for sure to a very fast vehicle, that goes often over 200 m/s. Probably we are dealing with an airplane.
    For this example a more suitable quantity to consider could be directly the speed and its distribution, in fact, the percentiles approach will be used also with other quantities.
    As we will see later, dealing with feature importances, the predictive power of the spacedelta percentiles probably lies in the subway detection. This transport mode runs part of this trip underground, for this reason the classifier will find multiple points that are much distanced with respect to average. A rule implicitly learnt by XGboost will be that if a trip contains only short distanced points, probably is not a subway trip. From a quantitative point of view, if the 95% percentile of spacedelta is low, it means that 95% of segments are short, then, the transport mode is not occurring underground.
- **Duration**: of the trip is the time occurred between the first and the last sample of a trip.
- **Average speed:** is the total distance of the trip divided by its total duration.
  - **Speed percentiles**: as for spacedelta, also for the speed of the various segment the five percentiles are computed, internally to each trip, and used as features. We will see how the 95% percentile of speed, comparable with the maximum velocity, but more robust to outliers, is one of the features with the highest importance according to different metrics. In fact, also intuitively, the maximum speed is the characteristic that best distinguish a mode from another. It remain still open the issue of regulations: this feature always work to discern a muscle propelled vehicle from a motorized one, but among this last group the maximum speed strongly depends on the local laws, and so the analyzed city.
- **Average acceleration**: considering the accelerations assigned to each point of one trip as point-level features, it is the mean of those accelerations.
  - **Acceleration skewness**: considering the empirical distribution of the above mentioned point-level accelerations, for each trip is possible to compute the skewness. It is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.
  - **Acceleration kurtosis**: From the same distribution mentioned above the kurtosis is measured as an indicator of the “peakedness”.
  - **Acceleration percentiles**: as before, also for acceleration, five percentiles are computed. Of course there is the possibility of redundancy with previous features, because the combination of percentiles is able to inform both about the asymmetry and the peakedness of the distribution.
- **Average angle**: is the average of all the angles contained in a trajectory (the sequence of segments in a trip), as defined in the previous paragraph.
  - **Angle skewness**: indicator of asymmetry of the angles distribution.
  - **Angle kurtosis**: indicator of “peakedness” of the angles distribution.
  - **Angle percentiles**: are the usual five percentiles computed from the angle distribution of the trajectory. The importance of the angle percentiles lies in the ability to describe how many times a user changes direction, and intuitively that happens very often while walking and very few for train trips, constrained by the path, that for technical reasons should be as straight as possible.

The second group of trip-level features is linked to the presence of POI (points of interest) along the trajectory. In this case, only transit stations are considered, but surely, including other amenities can improve the model performances.

For train stations:

- **nr_tr_st**: is the number of unique “touched train stations” along the trip. A station is considered touched if at least one datapoint of the trajectory falls in the range of 300 m from

the centroid of the station. As mentioned before, the dataset of Chinese train stations is obtained querying OpenStreetMap through Overpass-turbo.
- **nr_pti_tr_st**: counts the absolute number of points of the trip that falls in the range of 300 m from the station.
- **tr_touch_div_km**: I haven't found for this indicator an intuitive explanation. Adding it by mistake I noticed that improved the recall for the class "train" from 69% to 80%, that is the number of train trips previously misclassified is now decreased. So I decided to keep this feature. It is computed as a ratio. The numerator is a boolean variable, it is 1 if the trajectory "touches" at least one train station, or 0 if this never happens. The denominator is the total distance covered along the trip.
- **tr_st_x_km**: is the number of unique train station touched per unit of length. It is computed as the ratio between nr_tr_st and the trip distance.
- **tr_st_x_sec**: is the number of unique train station touched per unit of time. It is computed as the ratio between nr_tr_st and the trip duration.
- **pti_tr_st_x_km**: is the number of trip datapoints "touching" a station per unit of length. It is computed as the ratio between nr_pti_tr_st and the trip distance.
- **pti_tr_st_x_sec**: is the number of trip datapoints "touching" a station per unit of time. It is computed as the ratio between nr_pti_tr_st and the trip duration.

Regarding subway stations:
- **nr_sub_st**: is the number of unique subway station touched along the trip. In this case a point of the trajectory is considered to "touch" a station if it falls in the range of 100 m from its centroid.
- **nr_pti_sub_st**: is the number of the trajectory datapoints that touch a subway station.
- **sub_touch_div_km**: as for trains, it is the ratio between a boolean variable and the trip distance. The boolean variable is 1 if the trajectory touches at least one station, 0 otherwise.
- **sub_st_x_km**: number of subway stations touched by the trajectory per unit of length.
- **sub_st_x_sec**: number of subway stations touched by the trajectory per unit of time.
- **pti_sub_st_x_km**: number of trajectory points touching a subway station per unit of length.
- **pti_sub_st_x_sec**: number of trajectory points touching a subway station per unit of time.

Regarding bus stops:
- **nr_bus_st**: is the number of unique bus stops touched along the trip. In this case a point of the trajectory is considered to "touch" a stop if it falls in the range of 100 m from its coordinates.
- **nr_pti_bus_st**: is the number of the trajectory datapoints that touch a bus stop.
- **bus_touch_div_km**: as before, it is the ratio between a boolean variable and the trip distance. The boolean variable is 1 if the trajectory touches at least one stop, 0 otherwise.
- **bus_st_x_km**: number of bus stops touched by the trajectory per unit of length.
- **bus_st_x_sec**: number of bus stops touched by the trajectory per unit of time.
- **pti_bus_st_x_km**: number of trajectory points touching a bus stop per unit of length.
- **pti_bus_st_x_sec**: number of trajectory points touching a bus stop per unit of time.

Regarding airports:
- **nr_air**: counts the number of airports touched by the trajectory. In this case, a point is considered to touch an airport if it falls inside the range of 500 m from the centroid of the building.
- **nr_pti_air**: counts the points of the trajectory that touch airports.

4.2.4 The importance of the distributions

The idea of considering the distributions of the physical quantities of the trips, comes from the intuition that different vehicles behave differently from a dynamical point of view. In this section we will focus on the statistical distributions of the selected quantities, of course we will loose the information about speed sequentiality. For example, if a car is standstill for 5 minutes and at 50 km/h for other 5 minutes, looking just at the speed distribution, will be indistinguishable from a

bus that alternatively, minute by minute, is moving (50 km/h) and and standing (0 km/h). This is an extreme case, indeed the following approach will be reliable enough to train the classifier.

Among the dimensions that describe motion, speed is probably the most important, we will see that it alone is able to correct classify most of the trips.
This seems particularly true looking simultaneously at the speed distributions belonging to all the different trips. In the picture below (Figure 5) we are observing 8527 unique distributions divided by transportation mode.
All of them seems to correctly fit the class they belong: *bike* and *walk* travelers spend most of their time at low velocity. These distributions are long tailed so there are some segments of the trajectories that are probably labelled wrong. For example, if a user walking to reach a train forget to declare the new transport mode, some *train* points will result as *walk*. Anyway, the mislabelled point are the minority: 24% of walk segments are above 7.5 km/h and just 6% are above 15 km/h.
In the first row of the table there are the tree classes that are most difficult to disentangle, for this reason other researchers merge them in one single class. They share the road infrastructure and so the same limits, but fortunately, the shape of their distributions display some differences: taxies seldom goes over 80 km/h with respect to cars, buses instead, go often much slower than the other two modes. In particular buses have an high percentage of point that correspond to standstill segments, probably linked to bus stops. In conclusion, tanks to the different services they provide, the presence of reserved lanes and the different paths they may prefer, different transport modes present different dynamics.
Regarding subway and trains, a very sharp pattern is visible, this is probably linked to the fact that rail vehicles tend to reach the cruise speed as fast as they can, and maintain it since the next section.

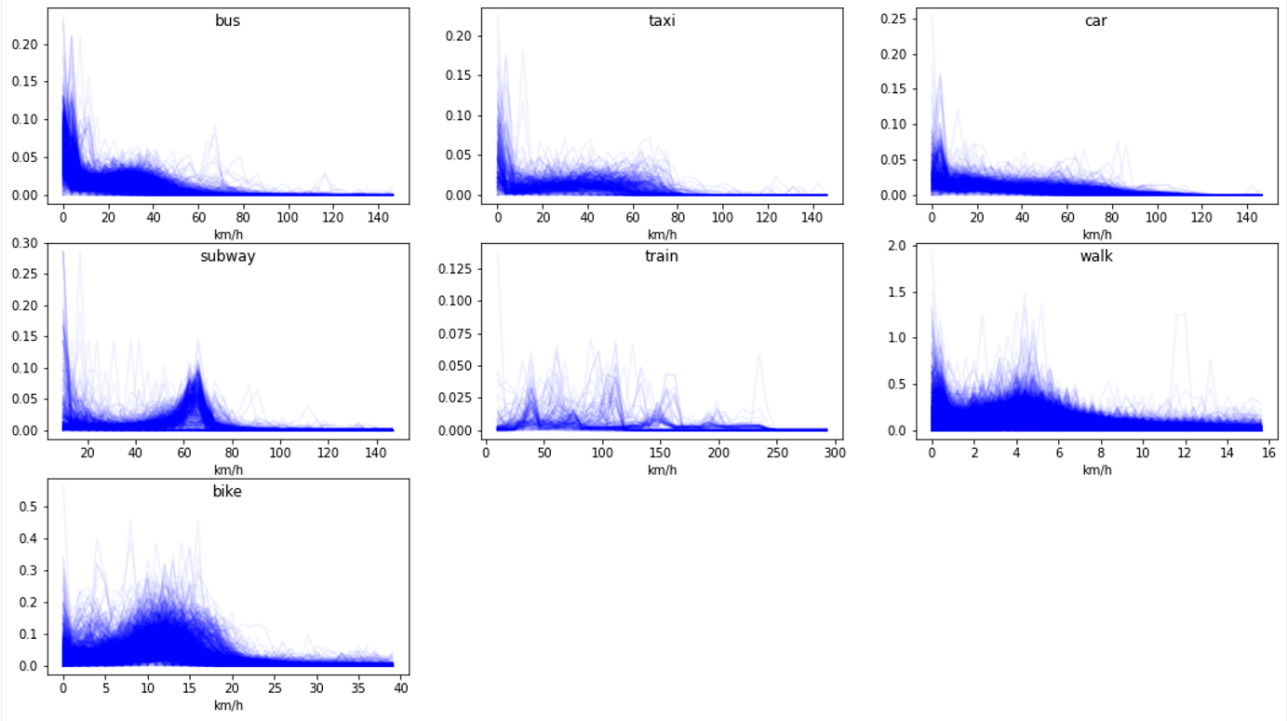


Figure 5: speed distributions of all trips, grouped by transport mode

5 percentiles can be seen as a very rough approximation of empirical Cumulative Distribution Function (eCDF), but still remain very informative with respect to individual statistics taken alone, such as mean, median, skewness, kurtosis. For this reason is it possible to combine all this features and push the model to its best performances.
The final evolution of this effort to provide the classifier many information about the various distributions, is to give it the distributions themselves.
This is achievable “rastering” the initial distribution in a finite number of bins, 40 for example. Before explaining it in detail, below, there is a picture of how it works (Figure 6).

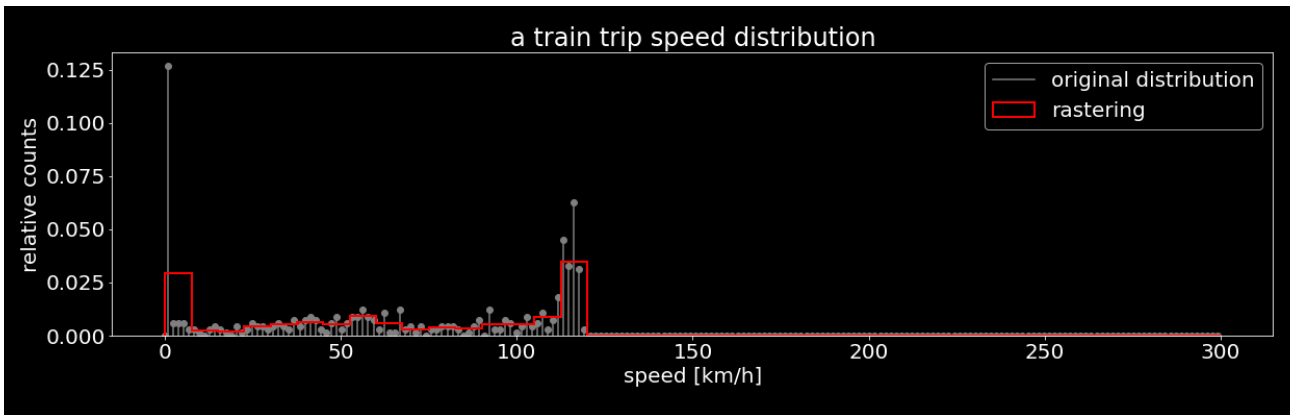


Figure 6: Example of the processing that leads from the original probability mass function (in grey) of a distribution to the “aggregated” one (in red)

Each bin will provide a feature in the following way.
Let’s take the speed distribution of the various segments inside a trip, and preserve only those which speed is between 0 and 300 km/h. Now, the histogram function from numpy is performed on the set of segment speeds, each value will increase the count of the relative bin. Having 40 equally spaced bins, the first one will count the number of segments for which speed is between 0 and 7.5 km/h, after this process is repeated for every bin, the total counts are normalized dividing by the sum.
The resulting dataframe has this aspect (Table 3).

tripID	3.75km/h	11.25km/h	18.75km/h	…	288.75km/h	296.25km/h	label
0	0.029530	0.002386	0.002088	…	0.0	0.0	train
1	0.018832	0.003766	0.001255	…	0.0	0.0	train

Table 3: Dataset preprocessed for training, the features are 40 and describe the mass density function of the distribution of speed for each trip.

Every row refers to a trip, the first 40 elements describe the **probability mass function** of its speed, and the last one is the corresponding ground truth.
Shuffling this dataframe, splitting it in 70% for train and 30% for test in a stratified fashion, I applied XGboost with the default hyperparameters. In the picture below some preliminary results are displayed (Figure 7).

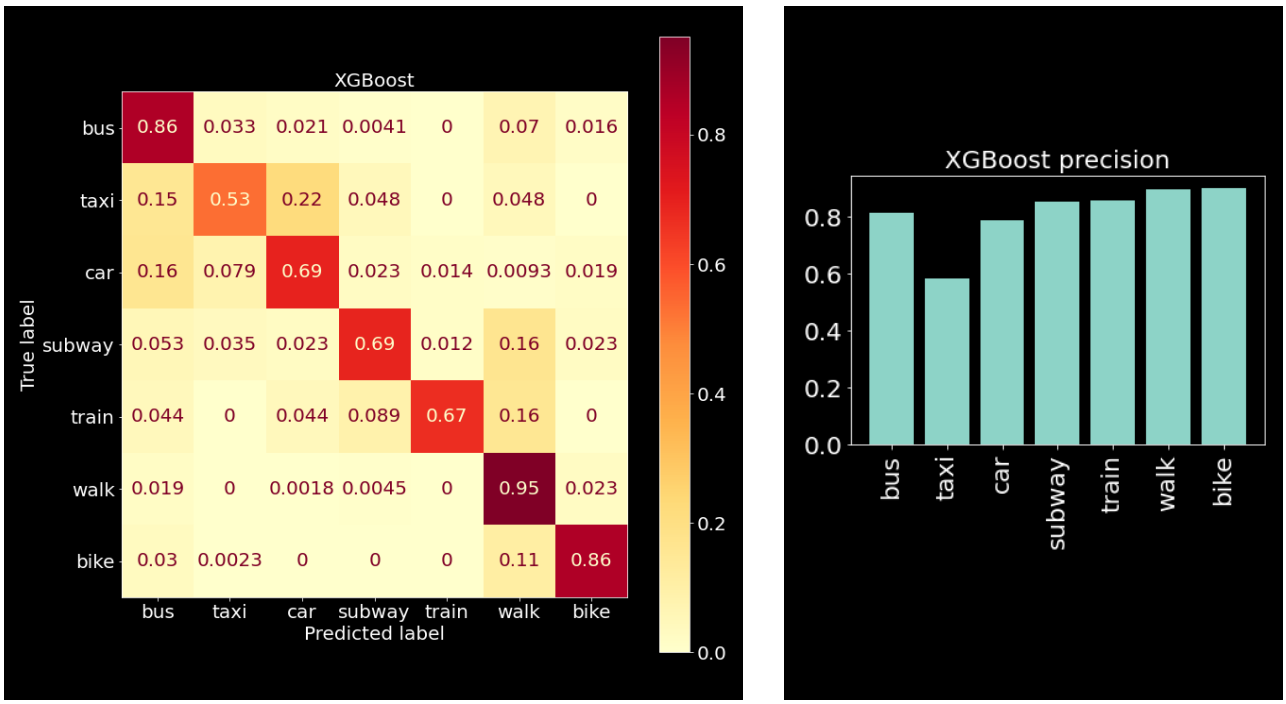


Figure 7: Metrics derived from an XGboost classifier trained only on an aggregated version of the speed distributions of the trips. On the left there is a confusion matrix, on the right the precision barplot.

A 10 fold cross validation is also performed, shuffling randomly and using a stratified train-test split, that is each set contains approximately the same percentage of samples of each target class as the complete set. The resulting average accuracy is 0.846 and the standard deviation 0.007.

These features will not be included in the final classifier, because they probably produce an overfitted model. Overfitting is linked to the fact that different countries have different speed limits, this is particularly true when dealing with railways. Since we are training the machine learning algorithm with the “most common speeds”, the resulting model will be well suited to detect a mode in Beijing but unreliable for trajectories sampled in other cities. I will deepen this topic in the *Future development* section.

4.2.5 Feature importances

XGboost is an ensemble method, that means that works building multiple decision trees and assigns a class to each trips through a majority vote.
The use of decision trees makes of it a method with an high explainability, in particular dealing with feature importances. Monitoring feature importances is an important task for multiple reasons:

- It allows to check if the intuitions during the feature engineering phase were correct.
- It provide rapidly an insight of features that have been built incorrectly, or if they contains errors.
- SHAP values are particularly useful in checking which features contributed the most to which class prediction.

In general, these values can serve both as a red flag and as a clue to discover unexpected behavior in a transport mode.
There exists multiple methods to compute feature importances, but all of them have pros, cons and can be biased [18].
Since the weaknesses of the different techniques are different, if a feature stands out according to all the indicators, it is probably a good predictor. For example “tr_st_x_km” contribute a lot according to *impurity reduction feature importance* but this is probably a bias of this metric, because the other metrics are in disagree.

For my work I used:

One model-specific method:

- **Default scikit-learn's** feature importance (Figure 8): is sometimes called "gini importance" or "mean decrease impurity" and is defined as the total decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)) averaged over all trees of the ensemble.[19]
  The pros are that is fast to compute and easy to retrieve (is a method of the classifier's object), the cons is that it's a biased approach. It has a tendency to inflate the importance of continuous features or high-cardinality categorical variables. [20]

Two model-agnostic methods, the first one compute global importances and the second one local importances. Agnostic means that the approach can be used to any type of classification algorithm. Local measures focus on the contribution of features for a specific prediction, whereas global measures take all predictions into account. [21]

- **Permutation** feature importances (Figure 8) directly measures feature importance by observing how random re-shuffling (thus preserving the distribution of the variable) of each predictor influences model performance.
  The the most relevant cons is that it overestimates the importance of correlated predictors [22]
- **SHAP value** [13] : is the average marginal contribution of a feature value across all the possible combinations of features. Every single predictions has 54 SHAP values, one for every feature, and each value tells how much that feature pushed in the direction of that prediction.
  The pros of this approach is that is possible to obtain a great overview both in a disaggregated and aggregated way. Aggregation can be performed in different ways, if these values are grouped by features, averaging on transport modes, it is possible to get a plot very similar to the other feature importances (Figure 9), if done by transport mode, but just visually, leaving all points disaggregated, it is possible to plot the so called "beeswarm" plot (Figure 10). The latter case is very informative and confirm us the reasonableness of the trained XGboost model.
  The cons is that it is built on a heavy mathematical background, but once overcame, it provides very intuitive explanations to the model outcomes. The other cons is that shap values do not evaluate the prediction itself, but just the path that leads to it.

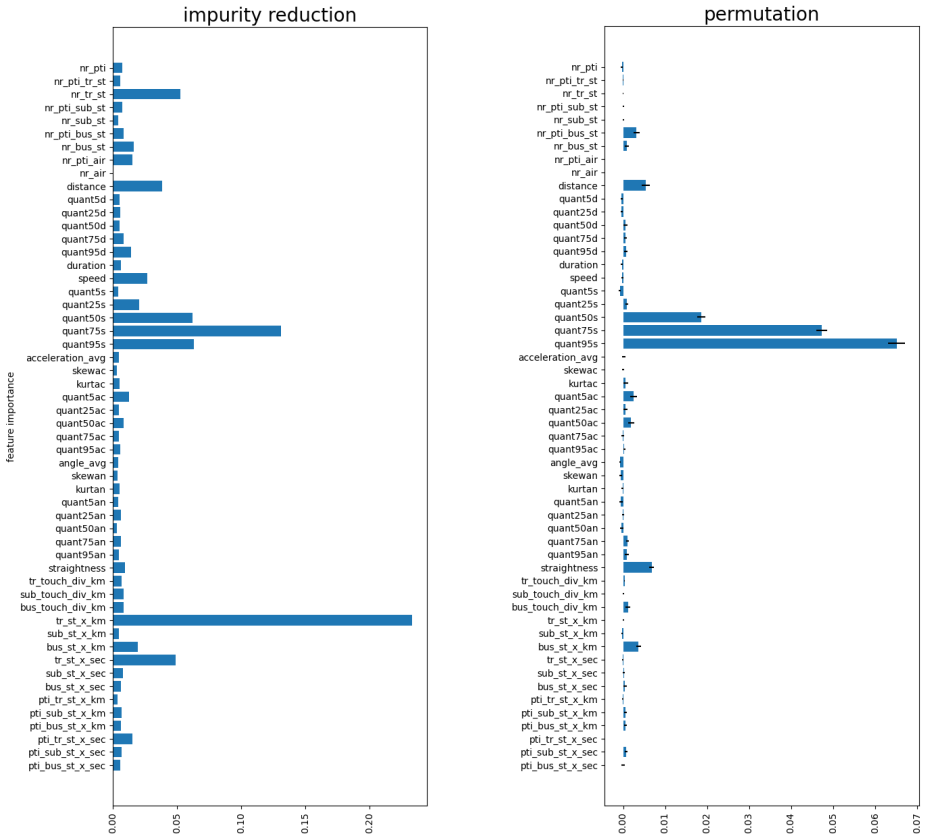


Figure 8: two types of feature importances, on the left the SKlearn's one based on impurity reduction, on the right the one based on permutations.

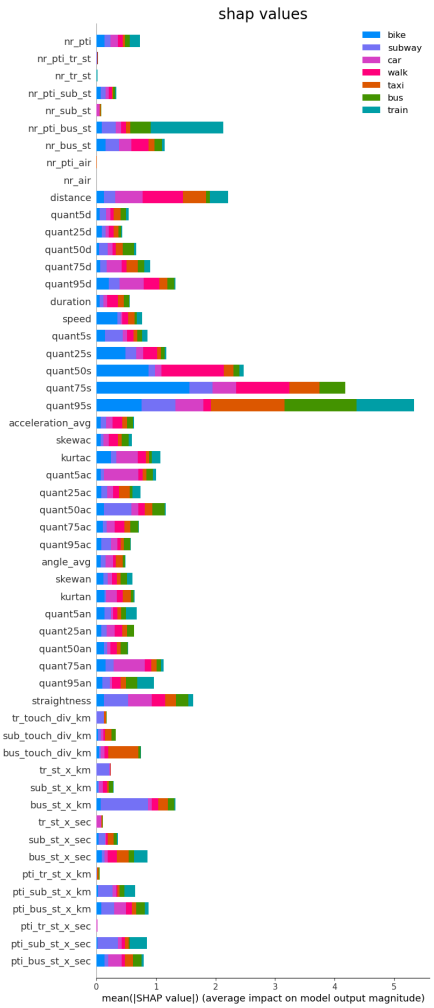


Figure 9: feature importances based on SHAP values.

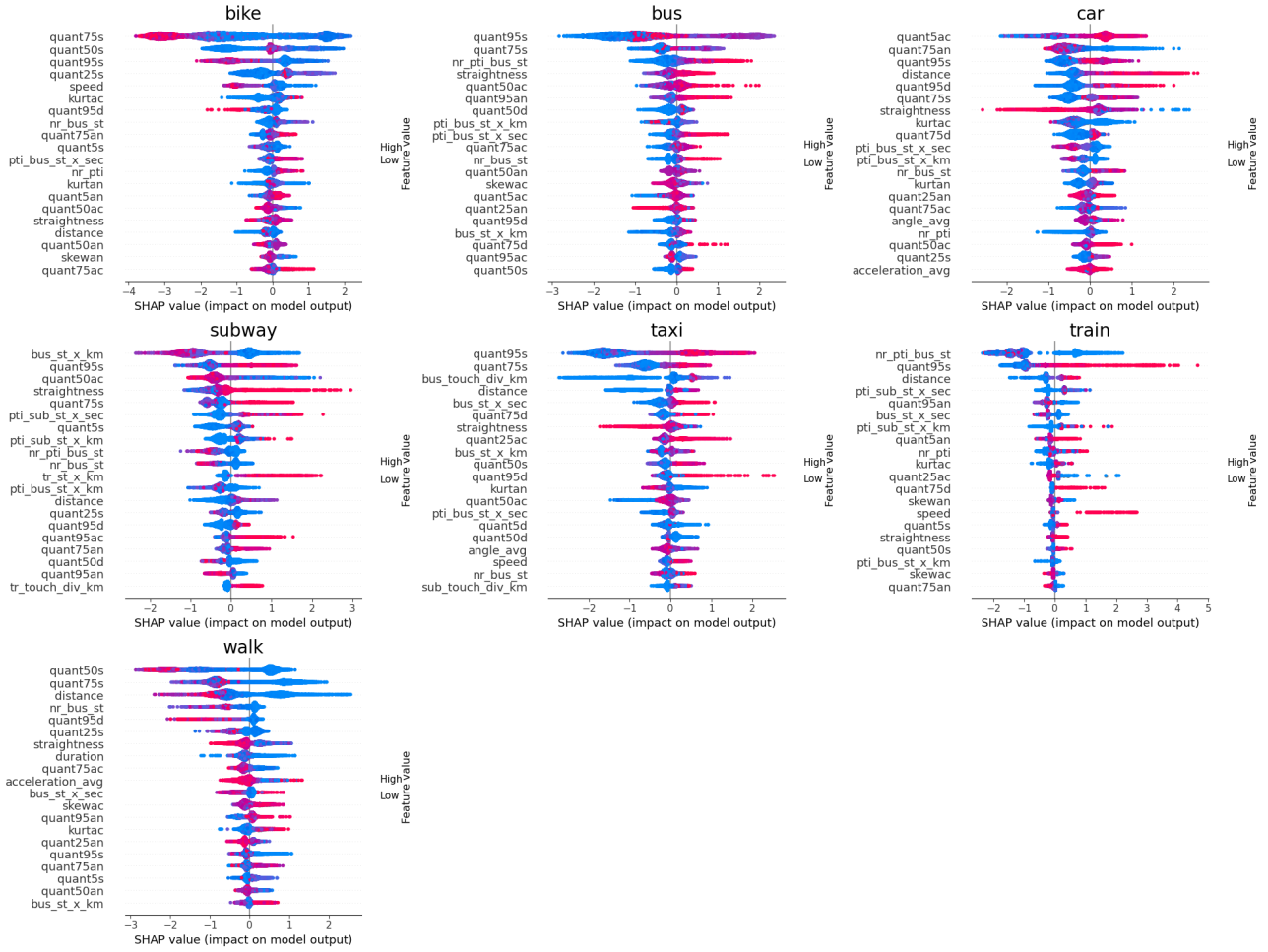


Figure 10: SHAP values beeswarm plots, one for every transport mode. The higher feature value, the more magenta is a point, the lower the feature value the more blue is a point.

To summarize, the most important features are the percentiles of speed (quant50s, quant75s, quant95s) according to all metrics, but SHAP values suggest that those are particularly powerful in detecting taxies and buses. Distance and straightness play also an important role. it's worth noticing how straightness impact differently if we are focusing on taxies and cars or on subways, for the first group, high value of this feature impact negatively, positively for the second class. In other words, it means that if a trajectory makes a lot of turns, probably, is not a car/taxi, but is very likely to be a subway trip.
Another interesting feature is the number of points lying in a bus stop, this feature is particularly powerful to predict both train and bus trips, but again in two different ways: if there are no bus stop along the trajectory, probably the user in on a train, otherwise he is in a bus.
This type of reasonings can be made for each case to corroborate a theory or to invalidate it.

4.3 Validation

In this part I will asses how the model performs on unseen data. The feature used are just those described in the feature engineering section. To check the validity of the classifier two different strategies have been performed: the first one uses **random train-test split** in a stratified fashion, the second is a "**leave one out**" strategy.

4.3.1 Random stratified 70%-30% split

Starting from the first approach, the dataset it's been divided into two subsets. To pursue the so called random and stratified sampling, the original dataset is divided in 7 "bins", one for each

class, and then, from each bin, 70% of the content of data points will populate the train set and the remaining 30% will populate the test set. In this way, the two sets will contains almost the same percentages of transport modes as the original source.
Once XGboost is trained, the resulting model is used on the test set to produce different evaluation metrics. The test set has at this point a list of trips and presents two different type of labels. The first label is the original one, the one declared by the user or the "True label". The second type is the "Predicted label" and is the one that the model assigned to that trip, according to the rules it implicitly learnt. A wide overview of the results is given by the confusion matrix (figure 11). Each entry of the matrix is computed in the following way:

$$entry_{i,j} = \frac{number\ of\ classes\ j\ predicted\ as\ i}{number\ of\ trips\ in\ class\ j}\ ;\ entry_{i,i} = \frac{true\ positives}{true\ positives + false\ negatives}$$

Where "*i*" is the index of the columns and "*j*" is the index of the rows.

An entry of the matrix contains the percentage of the trips originally labelled as "*i*" that have been interpreted as "*j*". According to this definition, if *i* and *j* coincides, the entry is the recall of the class *i*. So in an ideal case, if all the classes are correctly detected from the model, the confusion matrix would be an identity matrix, with the diagonal containing only ones.

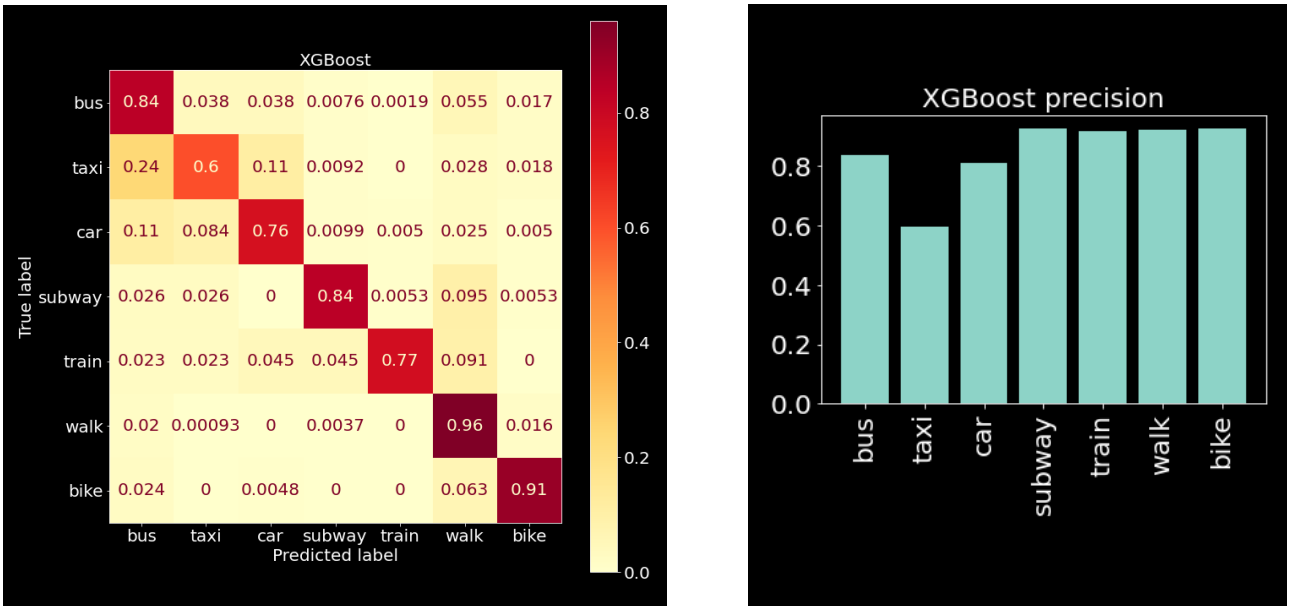


Figure 11: Metrics obtained from the 70-30 train-test split. On the left, the confusion matrix and on the right the precision barplot.

Among the classes, bikes and walk trips are almost always detected, rail vehicles are sometimes misclassified as "walk", lastly, there is a cluster in the upper left angle of the matrix that is composed by road motorized vehicles. This last group contains trips that are likely to be misclassified between each other.
To check the robustness of XGboost with respect to different train and test sets, a **10 fold cross validation** it's been performed. This method is similar to the previous one, the original dataset is divided into ten subsets of data, each containing the same amount of datapoints, sampled in a random and stratified way. One subset is used as test while the other 9 are merged together again and used for training, in this way the model is fitted and evaluated 10 times with different data.
The resulting **accuracy** has a mean value of **89,9%** and a standard deviation of 0,8%.
Looking just at the accuracy is not enough because it is unfair when there is a class unbalance. To overcome this distortion is useful to look at metrics in a disaggregated way. In fact confusion matrices and single class precisions are powerful tools in this field.
Also the precision plot is derived from the test set, it is computed, considering one class at a time, as the ratio between the true positives and the sum of true positives and false positives.

$$Precision_c = \frac{True\ Positives_c}{True\ Positives_c + False\ Positives_c}$$ where *c* is the class.

Depending on the final goal of the classification, it is possible to merge together some similar classes. For instance, considering only 5 classes (bus, car+taxi, subway+train, walk, bike) and performing a 10 fold cross validation, the accuracy increase up to 91,1% but are the single class precisions that have a greater improvement. After the aggregation, there isn't a class with a precision lower than 85%, while before, taxis were predicted with 59% of precision.

4.3.2 Leave one out

This second approach comes from the scarcity of users present in the Geolife experiment. Since the labelled dataset has just 64 users, and only few of them makes the majority of the trips, there is an high risk that the model is learning patterns from single recurrent trips and not the one of a transport mode in general.
A possible solution is the one of leaving just one user's trips outside the train set and use that user for testing.
In this case the model is fitted and evaluated 64 times, each time tested with a previously unseen user.
Here below (figure 12), the metric used to evaluate the classifier is precision. A disclaimer must be done before describing these plots. Previously, in the 70-30 split, the metrics described the performances of a single trained model. In this last case, the metrics are computed considering all the 64 different models simultaneously, in the following way.
The formulas to compute the entries of the matrix are still the same but the entire dataset is considered, since every slice it's been used as a train set. The True labels were already present, while the predicted labels were constantly assigned at each iteration. The same reasoning applies to precision score.

Predictably, performances strongly decreased, suggesting that the previous models were probably overfitting the data on the history of single users. If, from the training, we hide an entire set of trips belonging to a user, the model will be (in average) less powerful in detecting the modes of that person.
To deepen this issue, it's important to analyze the causes that leads to these poor predictions.
In the barplot below (figure 13) it's possible to visualize the accuracy for every single testing, since the users are few, It makes sense to search manually inside the the test sets that performed worse.
User 100 scores 0 accuracy because it contains only one trip, it's declared as "bus" but last only 8 seconds with a distance of 18 meters, so it's been reasonably misclassified as "walk".
User 89, contains 31 trips and accuracy is so low because all the trips are made by car but are often misclassified as the other two types of road vehicles (taxi and bus). This is a good news, because as mentioned before, our classifier is still able to recognize the macro area of road motorized vehicles. For the other users seems that the same pattern is repeated, sometimes classes are not detected, but the misclassification happens often in the same "macro class". That was already quantified in the confusion matrix (figure 12).
The other characteristic of the poorly performing users is the test set size: if a user contains few trips of a specific class, it seems more likely that these trips will not be correctly detected. To quantify this behavior of data, a scatter plot (figure 14) is built. Each point is represented, on the x-axis by the number of trips made by a single user with a specific transport mode and on the y-axis by the number of trips made by that user that are correctly predicted to use that mode of transport.
In the ideal case, all the predictions would be correct and the points would lie in the bisector y=x. In the real case we see a deviation from ideal one, in particular there are some points that goes below the bisector, but this effect tend to decrease with the increase of the test set size.
From this plot we are probably observing the human errors and the limit cases present in the test sets, because we have a "family" of points that follow the bisector and a "family" of outliers, the positive aspect concerning future developments in this field is that the outliers disappear when the test set increase. That means that with large enough dataset, the impact of limit cases and

wrong predictions is blurred by the effect of good predictions. Instead, for small test sets we can have both users containing poorer predictions and users containing perfect predictions.
There is only one user (ID=128) that, despite of containing 168 trips made by subway is almost never predicted well. Probably, this is due to a systematic phenomenon that cannot be mitigated by the high number of samples. Actually, all these trajectories follow a subway line, but, considering that 80% of the trips are done at a speed lower than 15 km/h and clusters of points lie close to subway stations, this suggest that the user starts declaring to be using the subway way before or after actually using it, but also that subway is a critical mode of transport that in future development of this work should be studied with more tailored features.

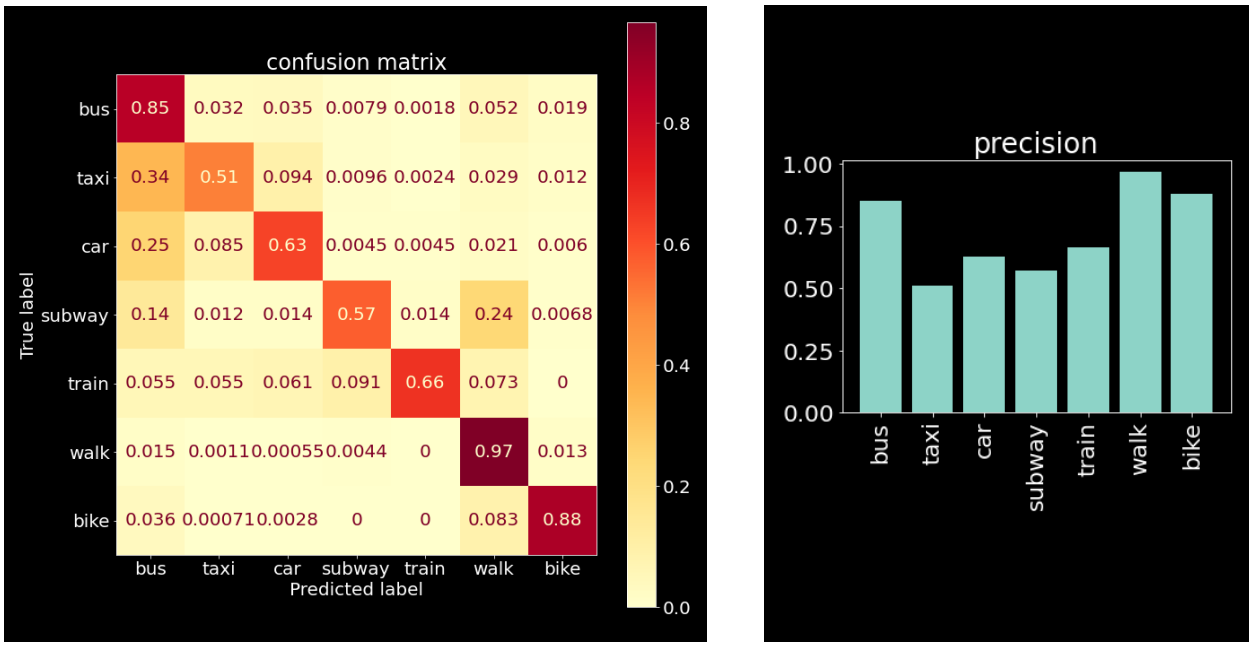


Figure 12: metrics obtained from the “leave one out” validation. On the left, the confusion matrix and on the right the precision barplot.

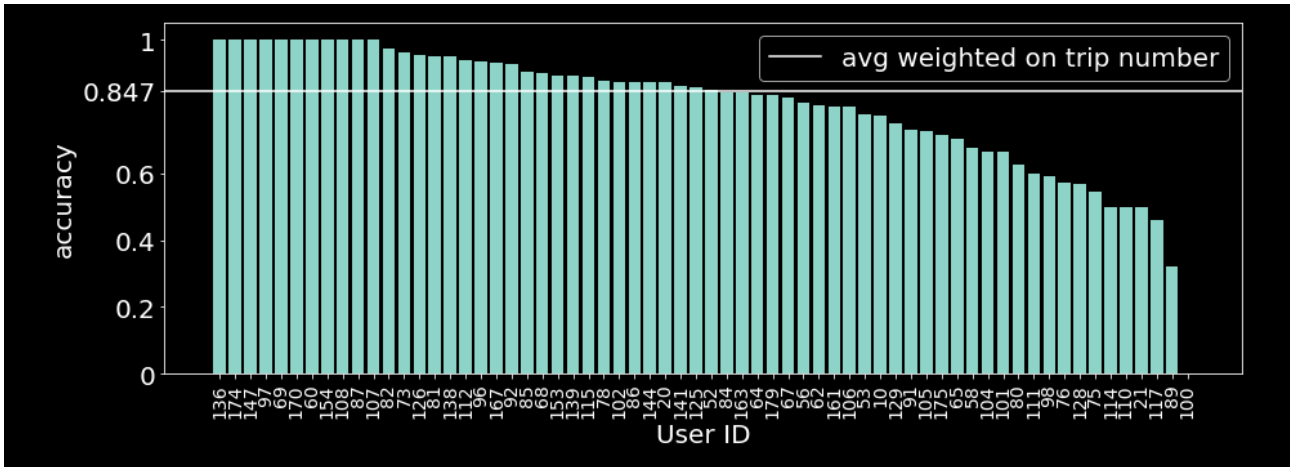


Figure 13: Metrics obtained from the “leave one out” validation. Each bar height is the accuracy related to a testing session, for which the test set is hidden in the training phase and contains only the trips made by a specific user. The white line is the average accuracy weighted on the number of trips made by each user.

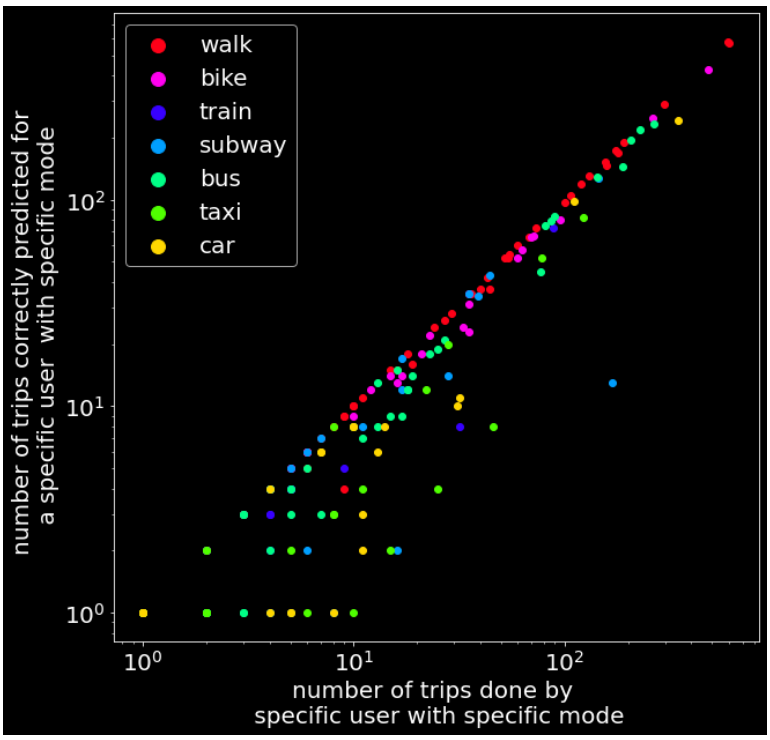


Figure 14: Scatter plot obtained from the “leave one out” validation. It describe the relation between the size of a test set and the number of correct predictions.

5. Conclusions and future developments

The final goal of this experiment was to confirm the possibility to detect transport modes starting from gps raw data and to provide Sony CSL, the research center under which I’ve done my internship, a trained classifier to label traffic flows from other datasets. Even if not fully developed, in particular in the last part concerning the extendibility to new cities, this work contains a guideline to built feature in such a way that the classifier accuracy can reach the state of the art.

The above mentioned Cuebiq dataset is a bit different from the Geolife one, It contains trips sampled in Italy, and the biggest limit is the low sampling rate. The good news is that resampling Geolife in order to emulate the low resolution of Cuebiq doesn’t degradate too much the performances. Accuracy is still high.
The Cuebiq dataset it’s been processed with a simple stop detector, if some subsequent points are distanced in time and space below a certain threshold they are labelled as stops, consequently the points that are in between stops are labelled as trip. Once trip are identified, it’s possible to compute the distance and the duration distributions.
This technique, together with the low sampling rate create a dissonance between Cuebiq and Geolife, in the way trips are extracted from a user’s entire history.
In Cuebiq most of the trips last several hours (figure 15), that because, the number of points is so scarse that stops or changes between modes cannot be detected. This, simply, is a problem that cannot be ignored and deserve a special treatment, because sparse trajectories are common in this field.

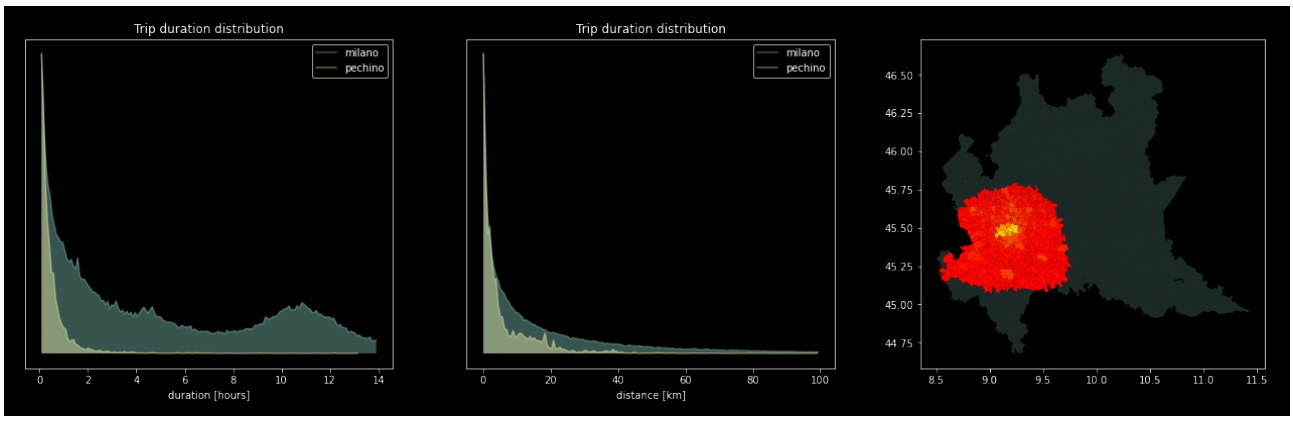


Figure 15: The plot on the left is a comparison between the trip duration distribution for the Beijing dataset (in yellow) and the Cuebiq dataset (in green). On the center the same comparison is performed for the distance. On the right the zones covered by Cuebiq data are highlighted, they bounded in such a way that contains Milan and some neighboring municipalities.

5.1 Trajectory smoothing

Some Cuebiq trajectories, instead present ah high density of points. Those are the one that most resemble Geolife ones. Even if they are a small fraction of the total ones, with a bit of preprocessing, the detection model can be applied. The problem of these trips is that they present a “shark tooth” behavior, for each coordinates, other two are sampled very close to it but a bit shifted in space and time. A moving average with a window of 4 points have demonstrated to mitigate this effect, creating more compatible distributions of motion features (figure 16, figure 17).

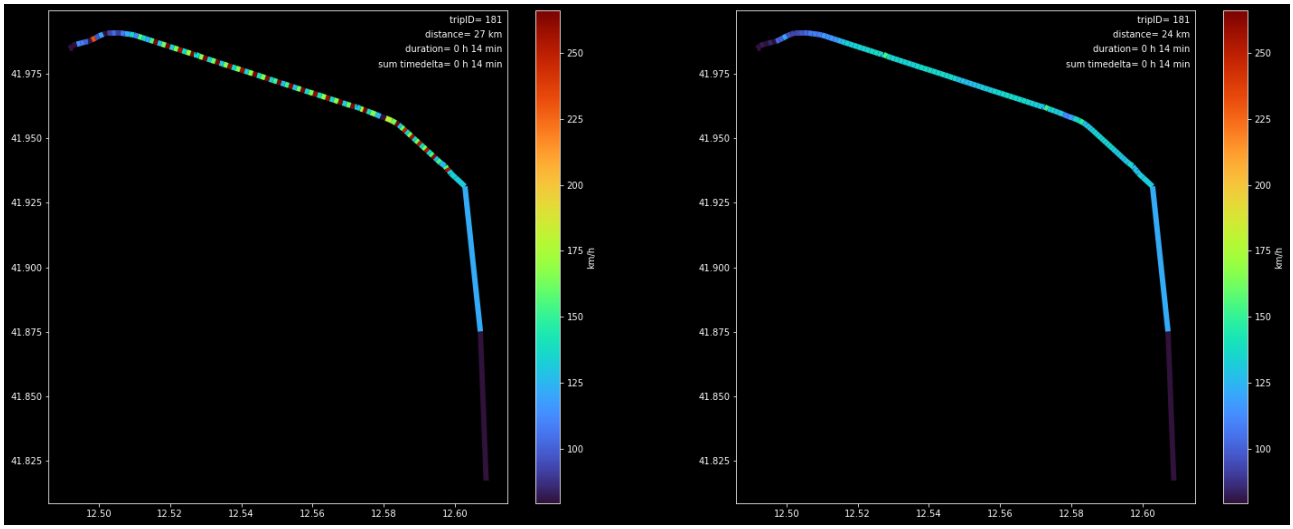


Figure 16: a trajectory before (left) and after (right) smoothing.

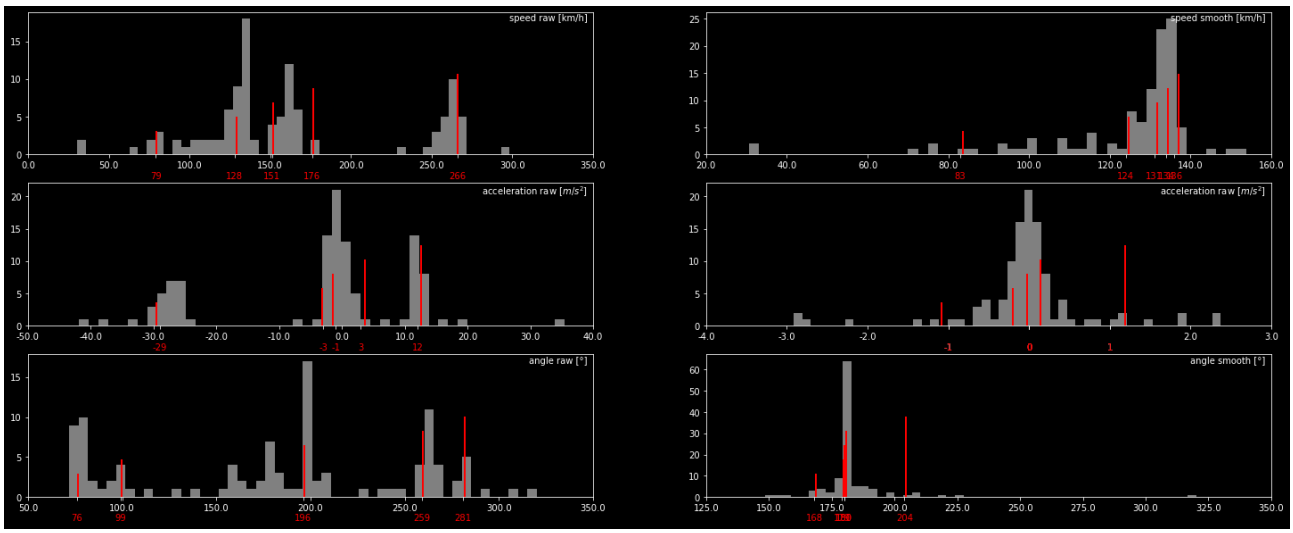


Figure 17: Motion features distributions before (left) and after (right) smoothing.

5.2 Future developments

There is still room for improvements to generalize the detection algorithm, even if the a ground truth to calibrate the model in other cities is missing. Assuming that Cuebiq users are representative of the actual population (it seems to be true for Lombardy) we could apply the trained model to label these new trajectories, count the aggregated flows for each transport mode and check the similarity with the official vehicular flows (coming from the OD matrix of Regione Lombardia).

A very rough estimation of the similarity between the predicted modes and the official one is provided in these two pie charts (figure 18). The “Chinese” classifier it’s been applied to a slice of the cuebiq database without any filtering, and the predicted labels have been counted (left chart), on the right side, the chart counts the official flows, aggregated by transport modes.

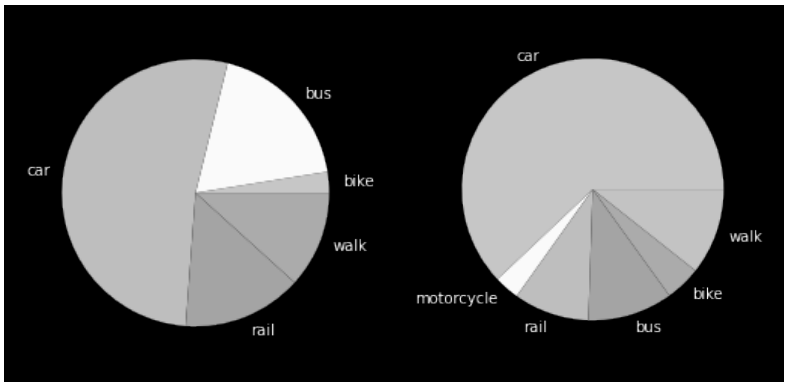


Figure 18: On the left a chart counting the transport modes detected from Cuebiq, on the right the chart counting the modes as presented in the official Lombardy OD matrix.

To conclude, there are different paths that can be taken, from merging together classes, to better trips splitting, to making the training Chinese set as similar as possible to unlabelled sets.

Bibliography

[1] WARDROP, John Glen. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1952, 1.3: 325-362.

[2] LOPES, André Soares; LOUREIRO, Carlos Felipe Grangeiro; VAN WEE, Bert. LUTI operational models review based on the proposition of an a priori ALUTI conceptual model. *Transport reviews*, 2019, 39.2: 204-225.

[3] BARBOSA, Hugo, et al. Human mobility: Models and applications. *Physics Reports*, 2018, 734: 1-74.

[4] MORO, Esteban, et al. Mobility patterns are associated with experienced income segregation in large US cities. *Nature communications*, 2021, 12.1: 1-10.

[5] LIMA, Antonio, et al. Understanding individual routing behaviour. *Journal of The Royal Society Interface*, 2016, 13.116: 20160021.

[6] https://www.dati.lombardia.it/Mobilit-e-trasporti/Matrice-OD2016-Passeggeri/tezw-ewgk

[7] ZHENG, Yu, et al. Understanding mobility based on GPS data. In: Proceedings of the 10th international conference on Ubiquitous computing. 2008. p. 312-321. (Geolife)

[8] https://www.cuebiq.com

[9] HUANG, Haosheng; CHENG, Yi; WEIBEL, Robert. Transport mode detection based on mobile phone network data: A systematic review. *Transportation Research Part C: Emerging Technologies*, 2019, 101: 297-312.

[10] XIAO, Zhibin, et al. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS International Journal of Geo-Information*, 2017, 6.2: 57.

[11] ETEMAD, Mohammad. Transportation modes classification using feature engineering. *arXiv preprint arXiv:1807.10876*, 2018.

[12] CHEN, Tianqi; GUESTRIN, Carlos. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016. p. 785-794.

[13] LUNDBERG, Scott M.; LEE, Su-In. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 2017, 30.

[14] ANDRADE, Thiago; CANCELA, Brais; GAMA, João. Discovering locations and habits from human mobility data. *Annals of Telecommunications*, 2020, 75.9: 505-521.

[15] PRISCO, Jacopo. Why UPS trucks (almost) never turn left. CNN, 2017. https://edition.cnn.com/2017/02/16/world/ups-trucks-no-left-turns/index.html

[16] https://overpass-turbo.eu

[17] https://www.openstreetmap.org

[18] LEE, Ceshine. Feature Importance Measures for Tree Models — Part I. Medium, 2017. https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3

[19] BREIMAN, Leo, et al. Classification and regression trees. Routledge, 2017.

[20] LEWINSON, Eryk. Explaining Feature Importance by example of a Random Forest. Medium, 2019. https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e

[21] STRINGER, Sven. Feature importance — what's in a name?. Medium, 2018. https://medium.com/bigdatarepublic/feature-importance-whats-in-a-name-79532e59eea3

[22] STROBL, Carolin, et al. Conditional variable importance for random forests. BMC bioinformatics, 2008, 9.1: 1-11.

Acknowledgements

La mia mente di rado mi concede il lusso di vivere il presente, convivo con una voce narrante che interroga il passato e scruta il prossimo futuro, non è necessariamente un'aspetto negativo, ma rende il "cogliere l'attimo" un momento ancora più prezioso. È per questo che voglio esprimere la mia gratitudine verso tutti. Tulle le persone che ho incontrato nella mia vita, la cui tangenza delle linee narrative ha prodotto stupore e ispirazione nella mia coscienza, ma anche e sopratutto l'idea di casa, come un focolare domestico dove rifugiarsi e da cui trarre forza quando fuori tempesta. Con queste persone ho condiviso momenti perfetti in sé, atemporali, lontani da quella routine frenetica e pianificatrice.
Queste persone sono il motivo per cui posso guardare il passato con nostalgia, vedere il futuro con speranza e vivere il presente con spensieratezza.
Grazie alla mia famiglia, ai miei parenti, agli amici, ai mentori, ai conoscenti, agli incontri fortuiti.