City Indicators for Geographical Transfer Learning: An Application to Crash Prediction

Mirco Nanni[1*], Riccardo Guidotti[2], Agnese Bonavita[3] and Omid Isfahani Alamdari[2]

[1*]ISTI, CNR, Via G. Moruzzi, 1, Pisa, 56127, Italy.
[2]Computer Science, University of Pisa, Largo B. Pontecorvo, 3, Pisa, 56127, Italy.
[3]Scuola Normale Superiore, Piazza dei Cavalieri, 7, Pisa, 56126, Italy.

*Corresponding author(s). E-mail(s): mirco.nanni@isti.cnr.it;
Contributing authors: riccardo.guidotti@unipi.it;
agnese.bonavita@sns.it; omid.isfahanialamdari@unipi.it;

Abstract

The massive and increasing availability of mobility data enables the study and the prediction of human mobility behavior and activities at various levels. In this paper, we tackle the problem of predicting the crash risk of a car driver in the long term. This is a very challenging task, requiring a deep knowledge of both the driver and their surroundings, yet it has several useful applications to public safety (e.g. by coaching high-risk drivers) and the insurance market (e.g. by adapting pricing to risk). We model each user with a data-driven approach based on a network representation of users' mobility. In addition, we represent the areas in which users moves through the definition of a wide set of city indicators that capture different aspects of the city. These indicators are based on human mobility and are automatically computed from a set of different data sources, including mobility traces and road networks. Through these city indicators we develop a geographical transfer learning approach for the crash risk task such that we can build effective predictive models for another area where labeled data is not available. Empirical results over real datasets show the superiority of our solution.

Keywords: Mobility Data Model, Crash Prediction, Individual Mobility Network, Mobility Data Mining, Car Insurance

1 Introduction

Collecting and processing mobility data is a fundamental task of car telematics and (modern) car insurance companies. Their main objective in doing that is typically to provide to end-users services like pay-as-you-drive contracts, anti-theft control, and prompt emergency rescue in case of accidents [1]. One of their foremost priorities, however, is to adapt policy pricing to customers in the best way, which mainly consists in finding a trade-off between profit and competitiveness. In this context, risk assessment is probably the most critical problem addressed. The risk from the company perspective can involve several aspects, yet the most impactful one is the customer's risk of having accidents in the future [2] since high-risk ones are likely to cause the company a loss (paying the costs of her accidents), while low-risk ones are more likely to provide a plain profit. In this context, since the car insurance markets are quickly expanding also towards new (for the market) geographical areas[1], there is the need to establish services in areas where very little or no prior knowledge at all is available, making the risk assessment task even more challenging.

Along the lines mentioned above, our research pursues two distinct objectives.

First, develop a methodology for **predicting the customer's risk score**: given a car insurance customer, provide a risk score relative to the long-term future, e.g., the next month or the next year. Since this estimate is expected to depend both on how the customer drives and on the conditions of the surrounding environment [3–5], we adopt an approach based on the computation of individual driving features, describing how much the user drives and how much dynamically, also related to the general characteristics of mobility in the places that the user visits. Since the raw mobility data collected by car telematics and car insurance companies is typically limited to positions and events of the vehicle [1], with no vision of what happens around it, our approach elaborates the data to infer higher-level knowledge, such as driving behaviors (frequent accelerations, average speed, etc.), individual mobility demand (detecting frequent trips, travel times during the day, etc.), habit changes, etc. [6]. That is achieved, in particular, by exploiting Individual Mobility Networks (IMNs) [6–8], a network-based representation that integrates important locations, movements, and their temporal dimension in a succinct way. Therefore, the proposed approach takes into account several different aspects: *individual* components of the driving behavior including those that can be derived from IMNs, elements considering the *collective* mobility of other users, and static *contextual* information such as road categories and the presence of points of interest.

The second objective, which is also the main focus of this paper, is to **enable the geographical transfer of crash prediction models**, i.e. to make the customer's risk score prediction system usable and effective also on areas where historical data about crashes is unavailable or too limited. Given

[1] https://tinyurl.com/32k589z2

an area where we want to asses the customers' risk scores and yet there is not a local training dataset to learn from, we derive a prediction model through techniques for *geographical transfer learning* which exploit the models and data available in other areas, in particular those similar to the one analyzed [9]. We define an array of geographical transfer learning strategies based on the data and the models available in certain areas that can be applied to target areas individually or as an ensemble. In particular, we rely on a set of city indicators [9] that can be retrieved for every area to evaluate the similarity between two or more areas. The measures considered covers a wide spectrum of features, thus providing a multi-perspective description of area area. They include a set of spatial concentration indexes of human activities; network features of intra-city traffic flows; mobility characteristics of the individual mobility, obtained from networks that represent the places and movement of single users; last, characteristics of road networks and how traffic is distributed in them. The city indicators allow to compare the different areas, using this similarity measure as a way to properly weight the contribution that each source area (i.e. areas where data are available and local models could be built) should give to the construction of a predictive model for the target area (i.e. the one where no data for training a model is available). The paper proposes several different strategies that exploit such weights in different ways, and provides an empirical comparison to find out the best one in terms of prediction performances. When comparing models, performances are an important aspect to consider, but not the only one. Indeed, two models might have a similar accuracy, and yet implement completely different logics, for instance considering completely disjoint subsets of features. In the experimental section of this work we aim to understand in depth in what aspects the different models actually differ, and we realize that through the adoption of explainable AI approaches. That allows us to provide some hints about the reasons why the transfer of the models trained on certain areas and applied to a certain target area works better than in other cases.

We evaluate the proposed methodology on three datasets of real cars moving in three different areas, namely two cities (Rome and London), and one region (Tuscany, Italy). In particular, a deep study on the models' transferability is performed on the Tuscany dataset working at the province level, which provided a good variability of city contexts yet involving areas of comparable complexity. The results show that the individual mobility-based and context-aware modeling of the users that we propose improves the performance over the baselines that adopt state-of-art features. These results support the importance of the heavy feature engineering proposed in the paper to adequately solve the crash prediction problem. Finally, we observe that the best results in geographical transfer learning are obtained by the solutions based on the city indicators for training the most adequate classifier in a certain area. The explanation of these transferred models with SHAP reveals that the most important aspects for the crash prediction on the transfers are related to

events that happens while driving towards regularly visited locations such as harsh accelerations or harsh cornerings.

To summarize, the novel contributions of the paper are the following:

- we expand the work in [6] on crash prediction, by studying how much the prediction span impacts on the performances and whether the feature engineering implemented in our approach can be replaced by a deep learning model over time series of basic mobility features (the answer being *no*);
- as follow up of the work in [9], we define the geographical transfer learning problem for a challenging task, namely individual, long-term crash prediction;
- we propose three multi-source geographical transfer learning strategies based on the city indicators introduced in [9], which are used to quantify the similarity of two geographical areas;
- we empirically evaluate our solutions against baselines and competing methods on a large real dataset of private vehicles. The evaluation includes a study of the features that characterize the different models, through explainable AI methods.

The rest of the paper is organized as follows. Section 2 summarizes the related works on crash prediction, transfer learning and city indicators. In Section 3 we formalize the problem definition and we recall concepts involved in the models designed in Section 4. Section 5 presents experiments in the form of a case study. Finally, Section 6 concludes the paper and discusses next challenges.

2 Related Work

In this section we report an overview of the most relevant works related to the three research areas involved in this paper: crash prediction, transfer learning and city indicators.

Crash Prediction. The literature on crash prediction is relatively large, studying car accidents from various perspectives, such as the risk of roads, the failure of safety devices or drivers' lack of attention. Yet, at the time of writing there are no works trying to exploit mobility data analysis and user modeling for crash prediction and risk assessment, with the only exception of [6]. A large part of the works focuses on real-time prediction of individual crashes, i.e., try to identify the events that lead to a crash in the next few seconds, thus providing feedbacks to the user as she drives [10]. Similarly, [11] developed a model for real-time collision detection at road intersections by mining collision patterns, while [4], using different data, tries to relate crashes to both behavioral characteristics and physiological parameters. Other approaches (e.g., [3, 12, 13]) work on identifying areas that show characteristics usually associated with accidents, such as increased traffic density, adverse weather conditions, etc. Besides features describing areas, the work in [14] also used individual vehicular data of cars (speed and time headway) passing through predefined detector

stations for improving the performance of a probabilistic model. In [15] it is presented a review of the key issues associated with crash-frequency data as well as strengths and weaknesses of similar methodological approaches. While extremely useful, such approaches result in being not applicable to fields like car insurance, where the focus is in creating a general risk profile of the user, thus implicitly involving the prediction of her crash risk in the long run, such as few months in the future. Only a few, preliminary works are available in this direction. In [2] machine learning methods are adopted to predict the users' driving behaviors, based on very basic movement statistics. The work in [6], which provides the starting point of the present paper, designs a data-driven model for predicting car drivers' risk of experiencing a crash based on the Individual Mobility Network model of the user and on statistical features which describe her driving characteristics. Here we extend the work and results of [6] with additional experimental studies and by boosting the crash prediction model with geographical transfer learning.

Geographical Transfer Learning. Individual mobility models and crash predictors, which are the basis of our proposed approach, are expected to strongly dependent on the specific geographical area under study. For instance, it has been empirically verified that the trip purpose classifiers in [7] work very well in the geographical area where they were extracted, but their performances dramatically degrade if applied to areas with different characteristics. Since some geographical areas could be insufficiently covered by data, due to the non-homogeneous penetration of tracking devices, it would be very difficult to build different models for different areas from scratch. A possible approach to the problem, then, is given by methodologies that make it possible to adapt models built in data-rich areas to less rich ones, which is basically a geographical instance of the general transfer learning problem [16, 17]. The transfer learning research area aims to transfer the knowledge available in one domain, called the *source domain*, to another one, called the *target domain* [18]. We refer to the particular case where the different domains are actually different geographical areas as *geographical transfer learning*. This specific topic is studied only sparsely in the literature, usually with objectives rather different from ours. The most common problem considered is image recognition, typically satellite image labeling, as in [19] and [20]. Both papers deal with deep learning classifiers that are requested to work on data-poor areas, and therefore the models learned in data-rich areas (usually CNN-based models) are adapted to the new domain. The authors of [21] focus on crime prediction and, again, try to exploit the knowledge available in some areas to make reliable predictions on a different one having too little data to build a model. Finally, [22] builds shared bike demand prediction models over some cities (especially large ones, where more data is generally available) and then adapt them to other (usually smaller) ones. The work in [23] shares some ideas with ours since it tackles the problem of labeling road networks and shows how assessing the similarity of street networks improves the transfer of a model from one city to another one. Our work tackles a more complex prediction problem, and compares areas

through a multi-dimensional view, yet our results confirm the general message of the cited paper. The methods we propose start from the paper [9], which exploited a set of descriptive features of cities to assess their similarities, studying whether the transfer of models across cities works better among similar ones. Both the prediction problems tackled and the model transfer method adopted were very simple. In this work, we expand those results considerably, considering a complex crash prediction problem and developing several more sophisticated model transfer strategies, yet still, exploit city similarities.

City Indicators. We conclude this section by briefly reporting the most important papers describing methods for characterizing urban spaces and defining city indicators, which will be used in our work to compare geographical areas. In this area, Geographical Information Science introduced several innovations that helped to automatize and extend an approach usually driven by a domain expert, including statistical methods for geography [24] and computational tools for managing large databases of information. City indicators have an important application in defining the sustainability characteristics of urban areas. Various attempts have been made to design indicators for monitoring sustainability at various levels, such as national [25] and city level [26]. As described in the review paper [27], the literature covers a wide range of aspects, including mobility-related ones (e.g., mobility space usage and functional diversity). However, very few attempts were made to systematically exploit big data sources to estimate them. One example was the Air Quality Now EU project [28], which used vehicular and public transport data to infer some measures. Yet, that is limited to direct and simple ones, such as traffic, speeds, and exposure to pollution. The literature also considers mobility indicators and road network properties as potential measures to adopt, which is aligned with our approach [29]. Finally, exploiting big mobility data to understand the properties of geographical spaces is a very active area [30, 31]. However, to the best of our knowledge, [9] is the only proposal where a wide set of complex indicators are collected in a systematic and reproducible way, directly aimed to make cities comparable in a computational way. Therefore, in our proposal, we aim at exploiting the approach and the city indicators defined in [9].

3 Setting The Stage

We introduce here the definitions of *trajectory* [32] and *individual mobility network* [7, 8], useful for understanding the rest of the paper and adapted to the approach proposed. After that, we formalize the car crash prediction problem in general and in the transfer learning setting.

Definition 1 (Trajectory) *A* trajectory *is a sequence* $t = \langle p_1, \ldots, p_n \rangle$ *of spatio-temporal points, each being a tuple* $p_i = (x_i, y_i, z_i)$ *that contains latitude* x_i*, longitude* y_i *and timestamp* z_i *of the point. The points of a trajectory are chronologically ordered, i.e.,* $\forall 1 \leq i < n : z_i < z_{i+1}$.

As additional notation, we refer to the i-th point of a trajectory t (namely, p_i) as $t[i]$, and to its number of points with $t.n$. Also, we indicate the longitude, latitude and timestamp components of point $t[i]$ respectively with the notation $t[i].x$, $t[i].y$, and $t[i].z$. We name *individual history* the set of trajectories that a user followed in a time period. More formally:

Definition 2 (Individual History) Given a user u, we define the *individual history* of u as the set of trajectories $H_u = \langle t_1, \ldots, t_n \rangle$ traveled by u. Also, we denote with $H_u^{[a,b]}$ the subset of trajectories of H_u that occur in time interval $[a, b]$, i.e. $H_u^{[a,b]} = \{t \in H_u \mid [t[1].z, t[t.n].z] \subseteq [a, b]\}$.

Individual Mobility Network

Given a user u, their associated history H_u can be processed to extract their *individual mobility network* (IMN) G_u. An IMN describes the individual mobility of a user through a graph representation of her locations and movements, grasping the relevant properties and removing unnecessary details.

Definition 3 (Individual Mobility Network) Given a user u, we indicate with $G_u = (L_u, M_u)$ her *individual mobility network*, where L_u is the set of nodes and M_u is the set of edges. Given an aggregation operator agg, for each node $l \in L_u$ we define the following functions:

- $\omega(l) =$ number of trips in H_u reaching location l;
- $\delta(l) = agg(\{\text{durations of stops in } l\})$;
- $\rho(l) = agg(\{\text{arrival times of trips reaching } l\})$;
- $\pi_t(l) = agg(\{\text{durations of trips reaching } l\})$;
- $\pi_d(l) = agg(\{\text{lengths of trips reaching } l\})$;

Operator agg can return either a single value (e.g. median) or a n-ple (e.g. average and standard deviation, or quartiles). The same functions are also defined on edges (movements) $m = (l_i, l_j) \in M_u$ in a similar way, this time considering only trips that start from l_i and reach l_j.

Nodes in L_u are locations that represent a group of stop points, and edges in M_u are movements that represent groups of similar trips between two locations. Given the individual history H_u, the IMN G_u is obtained by retrieving the locations L_u through a spatial clustering-based aggregation of stop points [33], and the movements M_u by grouping the trips between any pair of locations [8].

Problem Formulation

We define the *crash prediction problem* as the association of a user's probability of having an accident in the next time period with their recent historical mobility. The duration of the user's history to consider and of the next time period for which we make predictions are two fixed parameters. Reasonable durations for the context at hand will have the scale of one or more months.

Definition 4 (Crash Prediction and Risk Assessment) Given the *prediction time* τ_p, *history depth* τ_h and *prediction span* τ_s, we define the two time intervals $\bar{z_p} = [\tau_p - \tau_h, \tau_p]$, named *predictors interval*, and $\bar{z_t} = (\tau_p, \tau_p + \tau_s]$, named *target interval*. Then, the *crash prediction problem* consists in evaluating if user u will have a car crash during period $\bar{z_t}$ and what is the crash probability, based on the analysis of the user's mobility during period $\bar{z_p}$. More formally, we want to estimate:

$$p_{crash}(u) = P\left(u \text{ has crash in } \bar{z_t} \mid H_u^{\bar{z_p}}\right)$$

The period $\bar{z_p}$ is the knowledge we have about the user at the moment of assessing her risk, while $\bar{z_t}$ is where/the period when the crash to predict will or will not happen.

In a geographical transfer learning context, crash prediction has the same overall objective, yet the available information for estimating p_{crash} mainly comes from areas that are different from that of the user.

Definition 5 (Geographically Transferred Crash Prediction) Given a set $A = \{A_1, \ldots, A_n\}$ of n geographical areas, each associated to a set U_i of users, to a function $\pi^{(i)}$ that estimates p_{crash} within A_i $(1 \leq i \leq n)$, and to the training set H_u^{train} of each user used to infer $\pi^{(i)}$ $(u \in U_i, 1 \leq i \leq n)$; the predictors and target intervals $\bar{z_p}$ and $\bar{z_t}$; and an area $A^* \notin A$, associated to a set U^* of users; the *geographically transferred crash prediction problem* consists in computing the function π^* estimating the crash risk probability for each user $u \in U^*$:

$$\pi^*(u) = P\left(u \text{ has crash in } \bar{z_t} \mid H_u^{\bar{z_p}}, \left\{\pi^{(i)}\right\}_{1 \leq i \leq n}, \left\{H_v^{train}\right\}_{v \in U_i, 1 \leq i \leq n}\right)$$

The definition emphasizes the fact that the crash prediction function can use both the training data and the locally inferred models of the geographical areas in A, while for the area A^* we do not have access to a training dataset, the only information available being the data of the user in the predictors interval $H_u^{\bar{z_p}}$ $(u \in U^*)$.

4 Methodology

In this section we first show how it is possible to characterize a geographical area with mobility data driven indicators, following the work in [9]. Then, we present the methodology proposed in [6] for long-term crash risk prediction based on IMNs. Finally, we design a set of novel strategies for the geographical transfer of crash prediction models across different areas.

4.1 Defining City Indicators

The transfer learning approaches proposed in this work revolve around the idea that highly similar geographical areas can share data and models more easily. Therefore, it is critical to define an effective way to compute similarity scores between pairs of areas. We do that by defining a set of descriptive features,

called *city indicators*, for each area, and then compute similarities through standard metrics, such as the normalized Euclidean distance.

In this section, we briefly describe the families of city indicators we computed and adopted for our purposes. In this setting, we use the word "city" as a simplification, to generally refer to a geographical area (or geographical unit) which is not only the urban area of a city, but can also be a much larger one, like having the size of a municipality, a province or even a whole region. The city indicators are meant to provide a multilayered description of geographical units through quantitative measures, which have been selected among indexes adopted not only in traditional urban studies, but also mobility analytics and network science. Hence, they can provide a multifaceted view of the areas under study. As discussed in Section 2, such numerical descriptions of geographical units can have a wide spectrum of applications. In the following, we give to the reader an overview of the city indicators adopted. For the details of the complete list and a formalization we refer the reader to [9].

City Indicators

Given a geographical unit (or *city*) A, we define its associated set of city indicators $CI(A)$ as $CI(A) =< S_C(A), N_C(A), I_C(A), R_C(A) >$, where each element represents a set of features computed over the user mobility data and the street network of A, briefly defined below:

- *Spatial concentration indexes of human activities (S_C).* They answer the question "how does the density of people and activities vary across the area"? Examples of this indicators are *spatial entropy* [34], *Moran's measure* [35], and the *average nearest neighbor distance*. The extraction process exploits mobility data to infer stay locations, which are then used to approximate activity places and their distribution. These indexes help distinguishing areas where activities are concentrated in a small space against those where activities are well distributed over the territory.
- *Network features of intra-city traffic flows (N_C).* Each area is partitioned into a regular grid and then modeled as a network whose nodes are the grid cells, and edges connect cells whenever some users moved from one to the other. Nodes and edges are weighted according to the number of matching trips. By representing the geographical unit as a network it is possible to describe all the activities through network measures such as *node degrees* [36], *Louvain modularity* [37], and *interaction models* like gravitation [38] and radiation [39, 40].
- *Characteristics of the individual mobility (I_C).* Consider the mobility at the level of individual users. Then geographical units can be described by aggregated values of their inhabitants' mobility such as *average distance and duration per trip*, *average driving distance* and *duration per day*, *average amount of trips per day*. Also, an aggregation of the features of IMN can be used in this setting. For instance we can consider the *average size of the network*, the *average individual radius of gyration*, the *average individual modularity*, etc.

- *Characteristics of road networks and how traffic is distributed in them (R_C).* Consider the mobility at the level of roads. Modeling a geographical unit as a network where nodes represent road intersections and edges road segments, we have indicators like *amount of edges and nodes*, *amount of intersections*, *average node degree*; as well as a set of measures typical of complex network analysis such as *road network's closeness centrality* [41]. Moreover, through a combined analysis of mobility data and road structure, the traffic concentration is characterized by indicators of distribution skewness and concentration. The latter, for instance, allow to highlight areas where the traffic is concentrated in a small portion of the road network.

4.2 IMN-based Crash Risk Prediction

Our objective is to estimate the probability $p_{crash}(u)$ in the crash prediction problem definition. In this section we do that through approximation, along two steps: *(i)* first, the knowledge contained in $H_u^{\bar{z}_p}$ is represented through a set of meaningful yet (necessarily) lossy features, that will be discussed in details in the next sections; then, *(ii)* the probability function is learned through machine learning predictors.

Predictive Features

Each user u is represented by a vector of m features computed over her predictors interval, namely: $x_u^{\bar{z}_p} = \langle f_1, f_2, \ldots, f_m \rangle$. We denote with $X^{\bar{z}_p} = \langle x_1^{\bar{z}_p}, x_2^{\bar{z}_p}, \ldots, x_n^{\bar{z}_p} \rangle$ the matrix of n vectors describing the behavior of n users. We indicate with $y^{\bar{z}_t}$ the vector saying if a user has experienced a crash in the target interval $\bar{z}_t$, i.e., $y_u^{\bar{z}_t} = 1$ if user u had a car crash in period $\bar{z}_t$, $y_u^{\bar{z}_t} = 0$ otherwise.

Machine Learning Models

The matrix of features $X^{\bar{z}_p}$ and the vector of target values $y^{\bar{z}_t}$ are used to train a machine learning classifier, which yields as output a car crash predictor function $p_{crash}(\cdot)$. The crash predictor takes as input a vector $x_u^{\bar{z}'_p}$, describing user u's mobility in a given predictors interval $\bar{z}'_p$, and returns the probability she will have a crash in the corresponding target period $\bar{z}'_t$, based on the training performed on $X^{\bar{z}_p}$ and $y^{\bar{z}_t}$. As machine learning classifiers [42] we considered several possible options, including K-Nearest-Neighbors, Decision Trees, Support Vector Machines, Deep Neural Networks, Random Forests, LightGBM, etc. Indeed, any prediction model working on standard tabular data could be in principle applied, since the specificities of the data domain are already captured by the user's features $x_u^{\bar{z}_p}$. Through preliminary experiments, we decided to mainly focus on Random Forest (RF), Deep Neural Network (DNN), and LightGBM (LGBM), since they yielded the best and most stable results. The case studies in Section 5 are based on these models.

A secondary (yet very relevant) objective of our work is to find the possible factors that lead to a crash, whatever the nature of each factor, either causal

or simply correlated. In order to achieve that, we adopt three ways to infer the role played by each feature in the classification. The first one comes as a built-in feature of RFs, namely the *feature importance* score, which says how much important is overall a feature, though not describing if that is a positive or negative factor. The second way exploits recent results in the explainable AI domain, in particular, the SHAP method [43], which assigns the positive/negative impact of each feature on every single prediction allowing to make both single-user and collective considerations. The third approach consists in aggregating the absolute SHAP values of different predictive models, in order to compare them and get a glimpse of their differences in terms of *logics* followed, in addition to performances.

Predictive Features

A key component of the proposed approach consists in translating the raw mobility information contained in $H_u^{\tilde{z}_p}$ into a set of features $\langle f_1, \ldots, f_m \rangle$ able to capture its significant elements, and in particular, those useful for crash prediction. The following were computed:

- *Trajectory-based features.* These features include *position-based features*, containing classic indicators of trajectories, i.e., length, duration, speed; and *event-based features*, measuring characteristics of the acceleration- and direction-related events contained in the data.
- *IMN-based Mobility features.* These features adopt IMNs (introduced in Section 3) as higher level of aggregation of the user's mobility, to extract three different types of information: *(i)* the network properties of the IMN, *(ii)* mobility aggregates focused on high-frequency locations and movements, and *(iii)* temporal stability measures of the IMN.
- *Mobility Context features.* These features estimate contextual indicators by extracting collective aggregates from the history of all users in the dataset. Information like the number of events, average speed, and acceleration statistics are computed for each geographical section, and they are associated to the single user based on which sections they stopped in at least once, compute an average of each characteristic of the sections.

Details for each family of features are available in [6]. The features considered can be inferred from the basic information that any car telematics service is expected to provide, and in that sense provides a minimal solution that can be very easily adapted to work in different geographical areas. Where available, this set can be extended with other useful measures about details of accidents, physical features of roads (pavement quality, size, visibility, etc.), weather, and so on. Real applications that need to be fine-tuned over a specific geographical area could indeed benefit from other information layers that can be easily integrated into our solution as additional features. Considering such extra layers and studying their impact, however, goes beyond the scope of this paper, and is left as interesting future work.

4.3 Geographical Transfer of Crash Prediction Models

The basic idea of transfer learning is that the phenomena we want to capture (and that determine the value of the target variable to predict) are inherently present in other datasets, although in different proportions and maybe in different shapes. Therefore, the problem is to understand which parts of the data (in our case, which geographical areas) are more likely to contain cues and information useful to capture relevant phenomena, and thus exploit them for predictions. Hence, our objective for geographical transfer learning in crash prediction, is to explore ways for exploiting all the knowledge available on areas different from the target one, i.e., the one where we need a predictive model. With respect to the categorization presented in Section 2, we design a geographical transfer learning which is homogeneous (the data and the prediction tasks in the source and target domains are of the same type), multi-source (in general, we have several geographical areas with data we can exploit in the transfer) and transductive (we assume that labeled data is available in meaningful quantities only in the source domains).

The solutions proposed in this work try to overcome some of the main issues highlighted in [6] (and further confirmed in our experiments in Section 5). first, blindly applying a model from one region to another does not consider at any level the differences that the two areas might have. In our context, for instance, the road conditions in one area might require a different driving style than another one (reflected in the accelerations and contextual features), or the city size and traffic might impact the routine behaviors of users. Second, adopting standard weighting schemata based on feature distribution is possible only if rather significant data is available for the target domain, although unlabeled, which can be difficult in practical applications. In particular, in our reference insurance case study, the data is always associated with labels (crash or no-crash), the problem being instead to reach in a geographical region a sufficient mass of historical data. Also, since in our experiments we study the transfer between areas in the same region (Tuscany), it resulted that the differences between the features distributions are in most cases not significant. Third, the empirical studies in [6] focused on rather large areas. This leads to building models that are more generic, and therefore might not be able to capture local behaviors of smaller locations.

In the following, we introduce a few solutions based on the following principles:

- a good prediction model for an area can profit from the information (data or models) coming from other areas, the main open question being how to account for the differences;
- while each area might have its own local factors and patterns, driving and crash risk are expected to follow a common (potentially large and diversified) set of rules, although each area might adopt them in different proportions – total absence being mainly an exception;

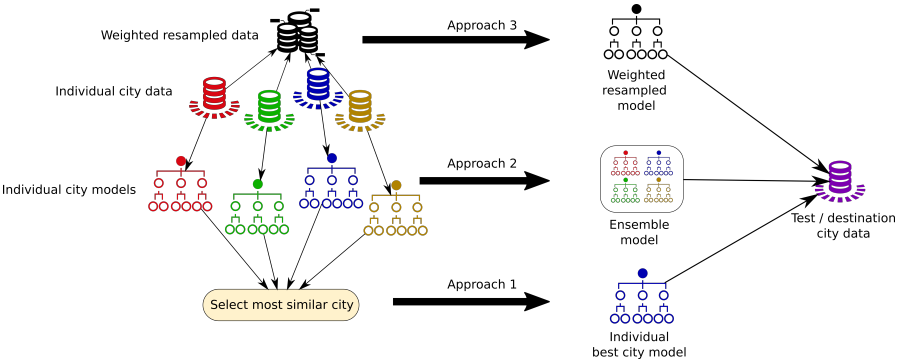

Fig. 1 Schema of the three geographical transfer learning approaches explored.

- the factors behind the events to predict, i.e., crashes, are strongly linked with the mobility context where the users move, therefore the city indicators described in Section 4.1 should provide a good basis for understanding how much two areas share the same type of context.

Based on these principles, we propose three approaches of varying complexity that follow them at different extents. Each solution is described in detail below, while a schematic summary is provided in Figure 1.

Approach 1: Best City Transfer

This is a direct application of the lessons learned in [6], namely that the model built on a city (or geographical unit) can be sometimes usable *as is* in another one, and that compliance is generally more likely to happen between cities that have similar spatial and mobility characteristics. Following this idea, Approach 1 selects among the source domains, i.e., the source cities where a model can be trained, the one that best matches the destination city in terms of city indicators, and applies its corresponding predictive model to the destination. With reference to Figure 1, the process starts from the individual city data, representing all possible source domains, over which we build individual city models. Finally, based on city indicators, we identify the source city that is most similar to the destination, and select its model. More formally:

$$p^{best}_{crash}(u) = p_k(u) \quad \text{with} \quad k = \arg\max_i sim(d, i) \tag{1}$$

where $p_i(u)$ is the crash probability of user u estimated by the individual model of source city i, and $sim(d, i)$ is the similarity between cities d (the destination) and i (the sources). More precisely, $sim(d, i)$ is computed from the Euclidean distance between the corresponding (normalized) city indicators of d and i, i.e.:

$$sim(d, i) = EuclidDist\left(z\text{–}score(CI(A_d)), (z\text{–}score(CI(A_i))\right)^{-1} \tag{2}$$

where *z−score* computes the attribute-by-attribute normalization of the city indicators.

We name the model *individual best city model* (bottom line of Figure 1).

Approach 2: Weighted Ensemble Model

It extends the ideas used in Approach 1, considering that each individual city dataset brings not only information that is specific for that location, but also information of more general validity, that might apply to all cities or at least to a subset. That means that each individual city model might, in principle, highlight a pattern or rule of general validity that, for statistical reasons or noise in data, could not be spotted in other cities. The idea is, therefore, to combine together the knowledge brought by all the individual models in an ensemble fashion, i.e., a meta-model is built by combination of the single ones, and predictions are performed by a voting schema where every single model provides a prediction, and the collection of results are combined. Since more similar cities are more likely to share common rules, the models in the ensemble can be associated with a weight corresponding to the city indicators-based similarity. Also, since our models provide a crash probability, the single predictions are combined through a weighted average. Formally:

$$p_{crash}^{ensemble}(u) = \sum_{i=1}^{N} w_i \cdot p_i(u) \quad \text{with} \quad w_i = \frac{sim(d,i)}{\sum_k sim(d,k)} \tag{3}$$

As before, $sim(d,i)$ is the similarity between the destination city d and sources i, and $p_i(u)$ is the crash probability of u estimated by the local model of source city i. In Figure 1 this corresponds to the central arrow, which yields the *weighted ensemble model* (or simply *ensemble model*, if clear from the context) that is then applied to the destination city data.

Approach 3: Weighted Sampling

The ideas of the ensemble approach are applied here from a slightly different perspective. The ensemble model assumes that if the overall dataset contains a pattern or rule that is relevant for the destination city, then at least a subset of the individual models should be able to identify it, allowing the voting schema to bring it to the destination. However, that is expected to hold only for relatively strong rules, which can emerge from individual datasets, while that might not work for smaller patterns that leave many weak traces in the various datasets. Basically, the ensemble approach filters at the source weaker patterns, some of which might actually result to be significant overall. As possible counter-measure for this effect, Approach 3 creates an ensemble of datasets rather than models, i.e., it builds a representative dataset by a weighted sampling of all individual datasets. This combined dataset, then, is used to build a predictive model. Since, again, we expect to find more useful information in source cities that are similar to the destination, the sampling weights are

proportional to the city similarities. More formally:

$$p_{crash}^{resample}(u) = P\left(u \text{ has crash in } \bar{z}_t \mid H_u^{\bar{z}_p}, \{H_v^{train}\}_{v \in D}\right) \quad (4)$$

where D is the data sample built for destination d from sources A, and is defined as:

$$D = \bigcup_{A_i \in A} D_i \quad \text{with} \quad D_i \subseteq U_i \quad \text{s.t.} \quad |D_i| = N \cdot w_i \quad (5)$$

where U_i represents the set of users described in source city i, and N is the requested size of the sampled dataset, i.e., $N = |D|$. Weights w_i are computed as for Approach 2. The more complex form of Equation 4 highlights the fact that this approach requires learning a model from scratch rather than simply combining or selecting existing local ones.

In relation to existing generic transfer learning solutions, the first two approaches presented above provide a form of *relational-based* transfer learning, since the models built in one domain are used (possibly adapted) in the other; the last approach, instead, works through an instance weighting strategy, which belongs to the category of *instance-based* transfer learning [18]. In particular, the latter is close in principle to Domain Weighting [44], yet it relies on a higher-level notion of city similarity, rather than a comparison of features distribution – which might be difficult to implement if only little (unlabeled) data is available in the target domain, as it is expected to happen in our application scenario. Also, as already mentioned, depending on the spatial granularity, in some cases the attribute distributions might not vary significantly across geographical units, thus making it a weak criterion. Indeed, preliminary tests on the datasets adopted in our experiments (see Section 5.1) showed that the feature distributions over the provinces were rather similar, being statistically not clearly distinguishable at the level of single features (around 58% of province-vs-province comparisons over all features did not pass the Kolmogorov-Smirnov rejection test [45] with threshold 0.05), and obtaining PCA projections over the two largest principal components having visually almost identical distributions.

5 Experiments

In this section, we present a case study on two datasets of private cars in which we employ the proposed methodology[2]. We first introduce the datasets, and then summarize the results obtained on the crash prediction problem with and without geographical transfer learning, with a comparison between our solution and some baselines. We also extract explanations of the predictions returned

[2]The source code is available at: https://github.com/riccotti/CrashPrediction. The city indicators used in this paper can be obtained from the Track & Know project website (see next footnote), while the mobility datasets are proprietary, and cannot be publicly shared.

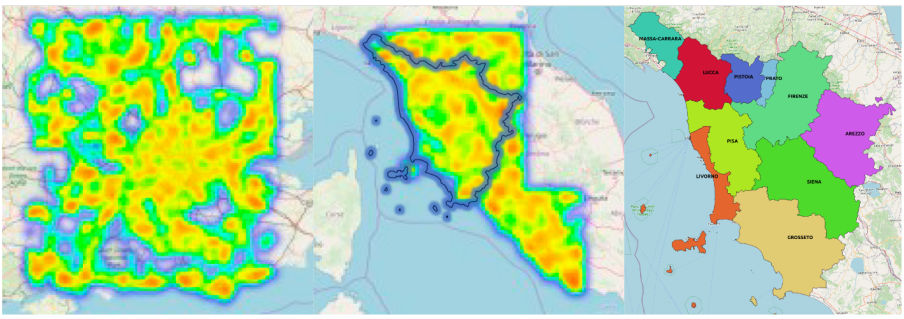

Fig. 2 Geographical areas of experiments. Dataset 1 includes London in UK (left), Tuscany and Rome in Italy (center). Dataset 2 is a zoom on the Tuscany area (highlighted in the center) by also considering its 10 provinces, shown on the right.

by the various models, and try to infer useful general hints for improving personal driving behaviors.

5.1 Dataset Description

The two datasets considered in our experiments consist of GPS traces of private vehicles tracked by a car telematics company and made accessible to us within the Track & Know project[3]. The first dataset, named Dataset 1, includes London in UK (Figure 2 left), Tuscany and Rome in Italy (Figure 2 center), each area having about 5,000 drivers. The second dataset, named Dataset 2, includes about 26,000 drivers and it is a zoom on the Tuscany area (highlighted in Figure 2 in the center) by also considering its administrative division into 10 provinces (Figure 2 right). We consider the partitioning of the Tuscany region in subareas in order use them as source and destination domains for transfer learning experiments. We decided to report results with respect to provinces because they provide a good trade-off between granularity and data availability on each partition. While testing model transfer across very different areas as Rome and London would be interesting, the different scale and complexity of these cities would require a more extensive dataset covering many other international cities, which was not possible in the scope of this work. In the rest of this section we will use the terms *city*, *geographical unit*, and *province* interchangeably, when there is no risk of confusion.

For both datasets, the raw mobility data consists of anonymized traces of vehicles of car insurance customers, containing the following information: *(i)* a list of GPS timestamped *positions* (latitude and longitude); *(ii)* a list of *events* in the form of timestamped position data enriched with labels describing events such as harsh acceleration, harsh braking and (possibly multiple) harsh cornering, with additional accelerometer metrics related to each event position. These data are collected whenever the accelerometer detects an acceleration

[3] https://trackandknowproject.eu/

exceeding predefined parameters; *(iii)* a list of *crashes* in form of timestamped position data related to crash events. Such events were originally detected through algorithms and later filtered by a human operator. The dataset is collected at an average rate of one position every 1.5 minutes, though there are some exceptions.

5.2 Experimental Settings

We organize the experimentation as follows. We use Dataset 1 to analyze the performance of the models for the basic car crash prediction problem, focusing the attention on the effect of the various features described in Section 4.2 and on the temporal dimension. On the other hand, we rely on the greater data availability of Dataset 2 to address the geographically transferred crash prediction problem with the city indicators described in Section 4.1 through the transfer learning methodologies illustrated in Section 4.3.

Local Crash Prediction. In the experimental setting for Dataset 1 (*D1*), we consider different time periods, corresponding to prediction times $\tau_p^1 = \textit{end of March}, \ldots, \tau_p^9 = \textit{end of November}$. The corresponding experiment periods $\bar{z}_i$ are obtained by fixing the history depth τ_h to 3 months (used to compute features) and prediction span to 1 month (the period where crashes are checked). We run the experiments in three different experimental settings, depending on how we consider the temporal and geographical components. In the first setting (*D1.1*) we keep separated each experiment period $\bar{z}_i$ and each spatial region r ($r \in \{London, Rome, Tuscany\}$) from all the others. In particular, for each given pair $(\bar{z}_i, r)$ we train a classifier over the corresponding data of all the users in r, namely $X^{\overline{z_{i,p}}}$ and $y^{\overline{z_{i,p}}}$, and then use the model to make predictions one month later, i.e., it is applied over $X^{\overline{z_{i+1,p}}}$ and the results are compared against the ground truth in $y^{\overline{z_{i+1,p}}}$. Notice that we must have $i+1 \leq 9$, therefore we obtain a total of $|\{\tau_p^i\}| \times |\{r\}| = 24$ sets of experimental results. In the second setting (*D1.2*), we still keep regions separated, while all experiment periods are considered together. Users are split into a training set and a test set, following a hold-out division[4], all the 9 experiment periods of a user in the training set are used (as 9 separate records) in the model training and, similarly, all the 9 experiment periods of a user in the test set are used for the model testing. The main difference between the two settings is that in (*D1.1*) we check if we can predict the crash of observed users in the future using a limited amount of data, while in (*D1.2*) we try to predict the crash of unobserved users using a consistent amount of data but without a temporal reference. Finally, the third setting (*D1.3*) amplifies the effects obtained by (*D1.2*) by putting the users of different areas in a unique training dataset.

Geographical Transfer Learning. The experimental setting for Dataset 2 (*D2*) is organized similarly to *(D1.2)*, i.e., geographical areas are kept separated, yet putting together all time periods. The main distinction is that now we have 10 areas corresponding to the provinces of Tuscany. In turn, each province will be selected as *target domain*, while all the others are used as

[4]Cross-validation was also tested, yet results do not change in any significant way.

Table 1 Datasets summary as average values of some features.

		% crash	tot traj	traj per day	tot evnt	evnt per day	nbr mov	nbr loc	degree
	London	1.08	280.54	3.39	2967	34.81	66.84	31.23	4.31
D1	Rome	2.82	307.48	3.13	2655	25.74	82.80	41.10	4.02
	Tuscany	3.12	327.11	3.28	3041	29.13	81.48	41.19	4.07
D2	Tuscany	0.84	375.41	3.92	1088	11.59	77.64	34.81	4.53

source domains, the task being to make predictions on the former using the models or data from the latter. The data related to each province is partitioned into a training and a test set, which are used to extract a local predictive model for each province, and then to test it on the other ones. The transfer learning approaches proposed will either select or combine such local models or build a training set by resampling the local training data, and then test the resulting model over the test partition of the province under analysis.

Datasets Preparation

In both experimental settings, before training the classifiers, we face two problems with the datasets analyzed. The first one is a class imbalance issue. Indeed there is a very low number of crashes compared to the number of no crashes (see Table 1). We tackle this problem by adopting the SMOTE oversampling approach [46]. The minority class is over-sampled by taking minority class samples and introducing synthetic examples along the line joining the k_{SMOTE} minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k_{SMOTE} nearest neighbors are randomly chosen. We adopt $k_{SMOTE} = 5$ by default as suggested in [46]. The effect of adopting SMOTE is to improve class balance and to reinforce the presence of the minority class in the decision regions where it appears. We highlight that we re-balance only the training datasets and not the test ones making the evaluation *harder* but more realistic. The second problem is the high dimensionality of the datasets analyzed. Indeed, the rich data engineering described in the previous sections leads to the construction of more than 400 features, some of them being highly correlated and redundant. This high dimensionality can cause difficulties in the learning of classification models. Thus, we adopt a dimensionality reduction technique based on correlation analysis. We calculated the Pearson correlation coefficient [47] between every pair of features for the various settings. Then, we removed one attribute for each couple having a correlation higher than 0.85. This operation reduced the dimensionality of the datasets to 162 features, with a balanced presence of trajectory-based, event-based, IMN-based, and contextual features. Table 1 reports the per-user average values of a small sample of features.

Machine Learning Models

Our crash prediction approach and our geographical transfer learning strategies can be in principle applied using any existing machine learning algorithm as an underlying predictive model. In this work, we consider three modern and

powerful types of classifiers: Random Forests (RF, basically an ensemble of several small decision trees), LightGBM (LGBM, a decision tree algorithm based on gradient boosting, with an emphasis on scalability) and Neural Networks (NN, here used in the simple form of a multi-layer perceptron).

Configuration details. For LGBM we used the lightgbm library[5], while for NN we experimented with both the Keras[6] and Scikit-Learn[7] libraries. Since the latter two libraries are applied to the same algorithm type (NN), and the models obtained with Keras yielded worse performances than Scikit-Learn, in the next sections we show only results for the latter. For all models we used the Randomized Search Cross Validation[8] to select the best combination of parameters. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings. For RF, we use the RandomForestClassifier that is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the "max samples" parameter if "bootstrap=True" (default), otherwise the whole dataset is used to build each tree. We try different settings to decide the number of trees in the forest (*'n estimators'*: [8, 16, 32, 64, 128, 256, 512, 1024]),the minimum number of samples required to split an internal node and the minimum number of samples required to be at a leaf node (*'min samples split'*:[2, 0.002, 0.01, 0.05, 0.1, 0.2], *'min samples leaf'*: [1, 0.001, 0.01, 0.05, 0.1, 0.2]). Final setting we adopted is the following:

- 'number of estimators': 128,
- 'min samples split': 0.05,
- 'min samples leaf': 0.05,

For NN we use the MLPClassifier, a Multi-layer Perceptron classifier that optimizes the log-loss function using stochastic gradient descent. Also in this case we tried different settings in order to find the optimal hidden layer size and the learning rate. We tried the 'relu', 'tanh' and 'logistic' functions as activation ones and we made experiments to try all configurations: 'hidden layer sizes': [(64, 128), (128, 256), (512, 1024), (512, 1024, 256), (1025, 512, 256)]. After testing, the final setting we adopted is the following:

- 'hidden layer sizes': (128, 256),
- 'activation function': 'relu',
- 'learning rate': 'constant',

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It has a high training speed and low memory usage. LightGBM uses the leaf-wise tree growth algorithm to get good results, and requires to select a few important parameters. The number of leaves (*num leaves*) is the

[5] https://lightgbm.readthedocs.io/en/latest/index.html
[6] https://scikit-learn.org/stable/
[7] https://keras.io/
[8] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

Table 2 Performance for the experimental setting *D1*. For D1.1 the metrics are aggregated in terms of means and standard deviation over different periods.

		Rome			Tuscany			London		
	Model	rec_1	$f1_1$	*auc*	rec_1	$f1_1$	*auc*	rec_1	$f1_1$	*auc*
D1.1	CST	1.00±.00	.024±.01	.500±.00	1.00±.00	.025±.01	.500±.00	1.00±.00	.009±.00	.500±.00
	RFI	.877±.10	.149±.08	.588±.05	.992±.01	.056±.04	.719±.05	.994±.01	.574±.02	.962±.01
	RFP	.891±.08	.140±.07	.574±.03	.992±.01	.042±.03	.577±.04	.719±.10	.308±.05	.612±.04
	RND	.486±.05	.352±.02	.500±.00	.488±.03	.355±.01	.500±.00	.499±.08	.341±.00	.500±.00
D1.2	CST	1.00	.028	.500	1.00	.029	.500	1.00	.010	.500
	RFI	.882	.216	.619	.944	.243	.775	1.00	.580	.955
	RFP	.866	.180	.586	.970	.061	.584	.624	.329	.574
	RND	.500	.361	.500	.480	.355	.500	.489	.344	.500

		All		
	Model	rec_1	$f1_1$	*auc*
D1.3	CST	1.00	.022	.500
	RFI	.991	.206	.776
	RFP	.996	.025	.641
	RND	.485	.352	.500

main parameter to control the complexity of the tree model. Theoretically, we can set $num\ leaves = 2^{max_{depth}}$ to obtain the same number of leaves as a depth-wise tree. However, this simple conversion is not good in practice. We tried to use $num\ leaves = (10, 31, 50)$ with a $max_{depth} = (-1, 2, 5, 10)$. The best parameters setting found is the following:

- 'number of leaves': 31,
- 'max depth': 5,
- 'boosting type': 'gbdt',

About the Keras experiments, we use the same configurations of MLPC Classifier with the only addition of the dropout parameter that is used to regularize the neurons activation and selection during the training phase. For our experiments we set 'dropout rate'=0.1.

Evaluation Measures

Given the application context around this work, our objective is to highlight future risky and potentially harmful events, also with the aim of raising an alarm that might help to prevent them. From this perspective, false positives are less critical than false negatives. To this aim we use as main evaluation guidelines [47] the *recall* of the positive class (rec_1), i.e., aiming to find as many risky drivers as possible, the f1-measure, i.e., the harmonic mean of precision and recall of the positive class weighted with respect to the number of crashes ($f1_1$), and the *area under the roc curve* (*auc*) of the positive class that is the area under the curve comparing the false positive rate (*FPR*) and true positive rate (*TPR*). All measures range from 0 to 1, the optimum being 1.

5.3 Crash Prediction Evaluation

In this section, we evaluate the results for the experimental settings in *D1*. Among the various classifiers, we found out that *Random forests* (*RF*) overcome those of the other algorithms. Thus, in the following, we report the

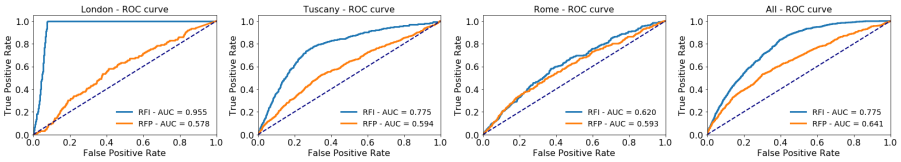

Fig. 3 ROC curve for different areas for *D1.2* and *D1.3*.

results obtained using RF classifiers[9]. We show the effectiveness of RF using the sophisticated IMN-based and contextual features described in Section 4.1 by comparing against three alternatives. The first two are baselines: a *constant* classifier (CST) always returning the positive class (crash); a *random* classifier (RND), predicting uniformly randomly crash or no-crash. The third one, instead, implements the approach in [2] with the information available in our dataset, adopting an RF only using the features from the state-of-the-art of crash prediction (RFP), such as average speed, number of breaks, etc. We name RFI the RF classifier that improves over RFP by extending the classical features used in literature with those we designed.

Table 2 reports the result for the experimental settings in *D1*, showing the evaluation measures returned by the classifiers for Rome, Tuscany, and London. Note that for the *D1.1* case the values are averaged among the various periods. The overall results we observe in the various experimental settings of *D1* are the following. The simultaneous analysis of the reported indicators shows that RFI provides the best and most reliable performances. Indeed, the CST baseline obviously has the highest recall but a zero precision on no crashes, making it useless for practical usage. On the other hand, RND easily gets a high $f1_1$, thanks to the high imbalance of data, but it loses half of the real crashes, with a recall below 0.5. RFP gives a better trade-off than CST and RND for the $f1_1$, but it shows an *auc* just slightly better than CST and RND, with a value around 0.6. On the other hand, RFI has always similar or larger $f1_1$ and recall compared to RFP, and it has systematically a higher *auc*[10].

In *D1.1* we observe different behaviors of RFI in the three areas considered. In London, RFI has the highest rec_1, $f1_1$, and *auc*. Notice that the other methods considered show much worse results. In other words, the new features introduced in this paper appear to make crashes easy to predict in London. Understanding the reasons for this effect is part of our future works. For *D1.2* we observe how the increased number of available records for the training leads to a not negligible improvement in the performance of the classifiers in the Rome, Tuscany, and London areas when compared to those of *D1.1*. In addition, the setting *D1.3* that puts together records from all the different areas ("All" section in Table 2) leads to a classifier even better than those resulting from *D1.2*. We highlight in Figure 3 the Receiver Operating Characteristic (ROC) curve of the classifiers for the experimental settings *D1.2* and *D1.3*. These plots show the evidence that London classifiers are much more accurate

[9] In particular, we used RF with 100 estimators, allowing leaves with at least 1% of the training data, and with a cost matrix weighting a crash 100 times more than a no crash.

than the others and that RFI classifiers markedly benefit from the usage of IMN-based and contextual features with respect to RFP, whose ROC curve is always below.

LSTM-based approaches

The key component of our approach that makes it superior to its closest competitor (RFP) is its extensive set of carefully engineered features, which are the result of a long experience in mobility analytics and driving behavior modeling. However, recent works in machine learning show that deep learning solutions are able to skip the human-made features construction phase in many tasks, and autonomously learn effective data representations directly from raw data, achieving exceptionally good performances. It is, therefore, natural to wonder if that can be the case also in the complex scenario we are considering. Along this line, we tested an alternative approach to our problem-based deep learning. In particular, we model the user's mobility as time series of basic mobility indicators, namely: maximum speed, distance covered, driving time and average trip duration. Then, we apply an LSTM network to learn the association between such time series and the target variable (crash / no-crash). The training and test data are partitioned exactly as in the experiments described above, and the time series has a 1-hour sampling rate. Experiments have been performed on Tuscany only since it is the richest dataset.

The network adopted follows the most commonly used structure for LSTM and time series classification: one LSTM level with 1024 units, followed by a drop-out of 0.5; then a dense layer with 256 nodes, followed by a drop-out of 0.2; finally, another dense layer with 64 nodes, and a drop-out of 0.01. In particular, the drop-out was necessary for the unbalance of the classes. A ReLu activation function was used in the internal layers, and a sigmoid function for the output. The training adopted an Adam optimizer with a binary cross-entropy loss function, using the area under the ROC curve (*auc*) as evaluation metrics. The misclassification weights were set to 0.5 for no-crashes and 95 for crashes, again due to the class unbalance. The preliminary results obtained, however, show rather poor performances. The *auc* has values close to random classification ($0.5 \pm .008$), and the *f1* measure is significantly lower than those obtained with the other methods ($0.01 \pm .005$). That is mainly caused by a low precision ($0.005 \pm .003$), whereas the recall is relatively good ($0.66 \pm .491$) yet rather unstable and lower than the other methods. Our conclusions are, therefore, that the approach, although interesting and worth exploring, does not work well with the basic features and the standard setting adopted, and further investigations are needed. We point them out as possible future works of this paper.

Testing longer prediction spans

An interesting aspect to study is whether predicting crashes over a longer time horizon is harder or actually simpler. Indeed, on the one hand we are trying to infer events that are further in the future, and therefore harder to

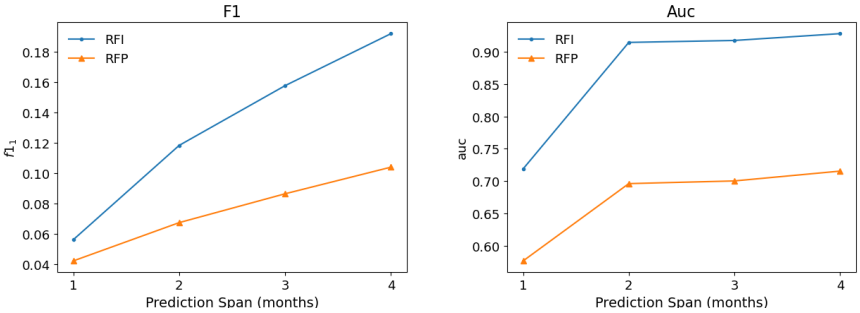

Fig. 4 *F1* score and *auc* for the RFI and RFP approaches on the Tuscany dataset by varying the prediction span from 1 month to 4 months.

Table 3 Crash prediction performance for the various geographical units inside Tuscany in *D2*. Each model is trained and tested in the same area similarly to *D1.2*. The last line report the performance of a model trained and tested on the whole dataset similarly to *D1.3*.

	RF			NN			LGBM		
City	rec_1	$f1_1$	*auc*	rec_1	$f1_1$	*auc*	rec_1	$f1_1$	*auc*
Arezzo	0.15	0.08	0.84	0.27	0.14	0.81	0.00	0.00	0.50
Florence	0.91	0.09	0.92	0.31	0.11	0.84	0.00	0.00	0.90
Grosseto	0.04	0.07	0.94	0.12	0.15	0.93	0.00	0.00	0.50
Livorno	0.83	0.10	0.90	0.00	0.00	0.97	0.00	0.00	0.50
Lucca	0.98	0.07	0.89	0.32	0.16	0.85	0.04	0.00	0.43
Massa	0.89	0.11	0.88	0.32	0.15	0.89	0.95	0.09	0.80
Pisa	0.53	0.12	0.92	0.31	0.26	0.85	0.00	0.00	0.10
Pistoia	0.31	0.06	0.83	0.40	0.07	0.85	0.00	0.00	0.50
Prato	0.35	0.25	0.91	0.45	0.21	0.94	0.00	0.00	0.91
Siena	0.36	0.20	0.86	0.36	0.34	0.93	0.00	0.00	0.50
All	0.44	0.12	0.91	0.34	0.11	0.96	0.46	0.07	0.83

capture; on the other hand, since we are enlarging the prediction window, and not just moving the same window further, the number of positive cases we are considering in the training phase is bound to increase, making the problem less unbalanced. In order to understand what is the resulting trade-off, we repeated the experiments made on the Tuscany area by changing the prediction span, now ranging from 1 month (the value used in the previous experiments) to 4, and measuring the *f1* and *auc* scores. The results are plotted in Figure 4, where also the values obtained by our main competitor RFP are given. In both cases, we can observe that longer spans are overall better captured by our models, meaning that the class unbalance is a stronger factor of the problem. We see, in particular, that while the *f1* score grows at an almost constant rate, the *auc* quickly reaches a sort of plateau, meaning that the associated risk probabilities produced by the model form a significantly better sorting when passing from 1 month to 2, yet no large improvement is given by further extending the window to 3 and 4 months. Interestingly, RFP follows exactly the same behavior, yet with much worse performances.

Table 4 Geographically transferred crash prediction *auc* for NN and RF. The best transfer are underlined, the transfer suggested by Approach 1 – Best City Transfer w.r.t the similarity of city indicators are in bold.

	NN *auc*										RF *auc*									
target → source ↓	Arezzo	Florence	Grosseto	Livorno	Lucca	Massa	Pisa	Pistoia	Prato	Siena	Arezzo	Florence	Grosseto	Livorno	Lucca	Massa	Pisa	Pistoia	Prato	Siena
Arezzo		.73	.73	**.80**	.81	**.81**	.73	.84	.80	**.79**		.82	.83	**.83**	.82	**.83**	.83	.82	.63	**.82**
Florence	.69		.80	.86	.89	.90	.85	.93	.90	.86	.93		.92	.93	.92	.92	.93	.92	.57	.83
Grosseto	.65	.87		.92	.82	.88	**.81**	.91	.90	.90	.92	.93		.92	.92	.93	**.91**	.90	.55	.84
Livorno	**.57**	.93	.78		.92	.93	.90	.97	.97	.89	**.96**	.97	.96		.97	.97	.97	.98	.49	.98
Lucca	.70	.72	.72	.87		.86	.74	**.89**	.85	.83	.88	.87	.89	.88		.87	.86	**.87**	.66	.86
Massa	.58	.78	.64	.80	.86		.81	.87	.83	.77	.88	.87	.87	.89	.89		.85	.86	.68	.74
Pisa	.46	.81	.71	.86	.89	.87		.88	.88	.83	.91	.90	.91	.91	.89	.89		.89	.68	.79
Pistoia	.62	**.63**	.60	.79	**.82**	.83	.79		**.82**	.80	.84	**.86**	.84	.86	**.86**	.86	.83		**.66**	.80
Prato	.69	.86	.69	.93	.85	.91	.73	.89		.85	.91	.90	.91	.92	.93	.89	.91	.91		.84
Siena	.74	.73	**.59**	.86	.90	.89	.86	.90	.87		.91	.88	**.90**	.92	.93	.92	.89	.91	.64	

Table 5 Geographically transferred crash prediction *auc* for NN and RF for the various approaches. Best results for each target area are highlighted in bold.

	NN *auc*					RF *auc*				
City	A0	A1	A2.1	A2.2	A3	A0	A1	A2.1	A2.2	A3
Arezzo	.546	.575	**.828**	.813	.813	.822	**.969**	.841	.841	.892
Florence	.501	.636	**.882**	.845	.849	.921	.864	**.928**	.915	.848
Grosseto	.645	.590	.849	**.931**	**.931**	.686	.908	.918	**.938**	.888
Livorno	.493	.803	.775	**.966**	.961	.885	.834	.885	**.896**	.863
Lucca	.451	.824	.842	**.847**	.808	.781	.861	.885	**.890**	.888
Massa	.602	.811	.852	**.887**	.886	.678	.836	**.890**	.885	.865
Pisa	.548	.818	.844	**.854**	**.854**	.877	.918	.898	**.920**	.868
Pistoia	.561	**.892**	.763	.847	.850	.728	**.872**	.864	.833	.811
Prato	.735	.823	.863	**.937**	**.937**	.843	.661	.905	**.906**	.860
Siena	.522	.799	.869	**.925**	.920	.783	.826	**.916**	.856	.686
Avg	.561	.756	.836	**.885**	.880	.800	.854	**.893**	.887	.847
Std	.08	.11	.03	**.05**	.05	.08	.08	**.02**	.03	.06

5.4 Geographically Transferred Crash Prediction Evaluation

In this section we evaluate the three geographical transfer learning strategies proposed in Section 4.3 in the experimental setting *(D2)*.

Testing local models

First, we analyze the performances of local models built separately on each province, applying them to the test set of the same area, similarly to what was done for setting *(D1.2)*. We adopt and compare the three predictive models described in Section 5.2: Random Forests (RF, the same used in *(D1)*), Deep Neural Networks (NN) and LightGBM. The results are summarized in Table 3, reporting recall, $f1_1$ and *auc* for each province and each algorithm. We can easily see that both RF and NN have high and stable performances, especially in terms of *auc*, which is the most informative measure. On the contrary, LGBM performs poorly in most provinces (7 out of 10), and is always worse than the other methods. This led us to focus the rest of the experiments only on RF and NN. The last line of Table 3 reports the performances obtained merging the data of all the provinces, thus building a unique global model and

testing it on all provinces. This is equivalent to setting *(D1.2)* on a different data sample or, from a different perspective, to setting *(D1.3)* at a smaller, regional scale. The results show performances that are perfectly aligned with the single provinces, suggesting that the larger training set of the global dataset is well balanced by the specificities of the local models of the provinces. In particular, this means that the local training data of provinces is sufficient to infer reasonable models.

A0: Baseline approach

The straightforward approach to exploit the data available in the source domains is to directly build a model using all the data, and try to apply it as is to the target domain. We experimented this approach as a *solution zero*, and its results are shown in Table 5, which will be used in the rest of this section as a reference for evaluating our proposed approaches A1-A3. As expected, this baseline results to be competitive with (though generally worse than) the simpler approaches (A1), and in most cases, significantly worse than the more sophisticated ones (A2-A3).

A1: Best City Transfer

Here we consider the first geographical transfer learning strategy we proposed, namely to make predictions on a target domain (i.e., the province under analysis) using a local model selected among the source domains (in our case, the 9 provinces left) by taking the province which is most similar to the target one. The results are summarized in Table 4, which reports the performances for all the pairs "source province vs. target province", marking in bold the values suggested by our first strategy. The performances are reported in terms of *auc*, and are shown for both the NN and RF algorithms. The values obtained suggest that the strategy works slightly better with RF, yet in general, it does not achieve satisfactory results, in most cases performing worse than the average. Apparently, single models do not provide knowledge which is directly usable, *as is*, in other areas, and then something more refined is needed.

A2: Weighted Ensemble Model

We test the second proposed approach, which consists of combining all the local (source) models into an ensemble, where their predictions over the target domain are aggregated. We compare our weighted combination, where each province votes with a weight proportional to its similarity w.r.t. the target, against a baseline where the weights are perfectly homogeneous. The baseline is named *A2.1*, while the weighted solution is named *A2.2*. Table 5 reports the results obtained for the two methods over each province, taken in turn as target domain, compared against the corresponding results of the best city transfer approach, named *A1*. Again, the results are shown bot for NN and RF, using *auc* as reference metrics, and highlighting in bold the best results. We can see that both *A2.1* and *A2.2* consistently improve over *A1*, thus confirming that combining the information of multiple sources is better than focusing only on

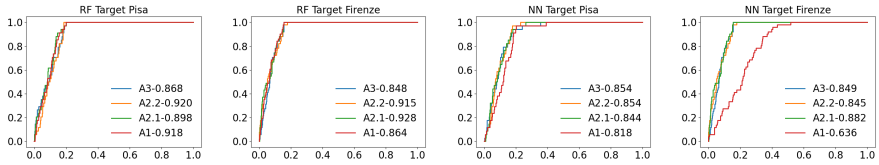

Fig. 5 Receiver Operating Characteristic (ROC) curve for geographically transferred crash prediction with target areas Pisa and Florence for *D2*.

one. At the same time, we can observe that *A2.2* performs overall much better than *A2.1*, especially with NN, proving that in this strategy, the similarity information becomes much more useful than what happened with the single-domain approach. Besides that, we can also notice that between NN and RF there is not a clear winner.

A3: Weighted Sampling

With the third strategy, we combine the local information of all (source) provinces at a lower level, combining data rather than models. As before, each province is considered in turn as target domain, yet this time we build a predictive model from scratch, obtaining the training data by sampling the training set of each source domain, taking larger samples from more similar provinces. The results are shown again in Table 5, under the column *A3*. Since the method involves a random sampling, the values shown are obtained as average over 10 distinct runs. The values point out that the strategy works relatively well in combination with NN, reaching very often performances equal or close to the best ones, yet providing overall slightly less convincing results (on average, there is a drop of 0.5% of *auc* w.r.t. *A2.2*). Also, the performances with RF are much worse since the average drop is 4%, and it never gets close to the best solutions.

An additional overall comparison of the results is provided by Figure 5, which shows the ROC curves of the models obtained with all four strategies discussed above, over two sample provinces: Pisa and Florence. In the case of A3, one of the 10 models generated was (randomly) selected. The plots show that in both cities, despite the differences in total *auc*, all strategies provide rather steep curves, and thus reasonable results, except for A1, which is less stable and, indeed, in the case of the NN predictor has significantly worse performances w.r.t. the others.

Conclusions on selecting the best transfer learning method

Summarizing the results seen above, we can conclude that combining the local knowledge of multiple sources is the key to improve performances in this transfer learning setting. This means, in particular, that using the baseline method A0 and the single-source method A1 is not recommended. In addition, the best level to perform such combination appears to be the weighted ensembling of local models (A2.2), rather than directly combining local datasets

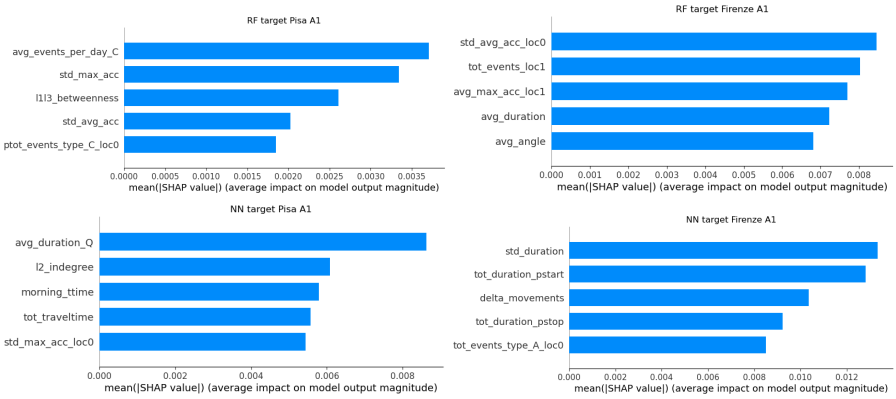

Fig. 6 Aggregated SHAP exlanation of the five most important features for geographically transferred crash prediction with target areas Pisa and Florence for *D2* using A1.

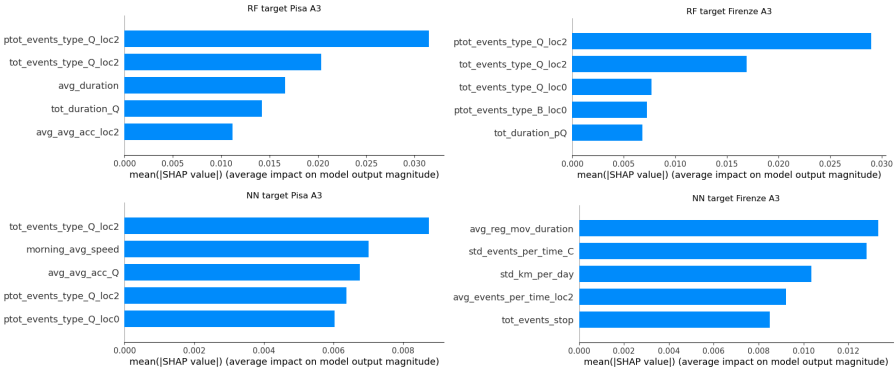

Fig. 7 Aggregated SHAP exlanation of the five most important features for geographically transferred crash prediction with target areas Pisa and Florence for *D2* using A3.

(A3), suggesting that in our data, the more detailed information that resampling strategies could in principle provide is outweighed by the noise that they introduce – noise that the local models have lost, together with other bits of (potentially useful) information. However, the data size and variability in different applications might change this equilibrium. Thus we suggest considering both approaches as reasonable candidates to test.

Geographically Transferred Crash Prediction Explanation

Like in [6], a parallel objective of this work is to understand which behaviors in a driver more likely could lead to future crashes. We realize it by adopting the SHapley Additive exPlanations (SHAP) method [43] to locally estimate for each prediction the expected contribution of each feature. SHAP returns the shapely values: the higher is a shapely value, the higher is the contribution of the feature; if the shapely value is positive, it contributes towards the positive class (crash); otherwise it contributes towards the negative class

(no crash). From [6] emerges that IMN-based features and collective features are fundamental for detecting crashes: the average maximum acceleration of break events in areas visited occasionally performed by other users is crucial in pushing towards the crash. Another feature having this effect is the number of acceleration and break events between the second and third most visited locations.

In the following, we summarize SHAP explanations by reporting the mean values of the absolute SHAP values for the drivers having a car crash. We focus our study on A1 and A3 to observe the differences between an approach trained on a single geographical unit (A1), and an approach trained on multiple weighted areas (A3). The idea is to understand which features are the most important for recognizing crashes in geographical transfer learning. The results are reported in Figure 6 for A1 and in Figure 7 for A3. We report the explanations for the records for both NN and RF, using Pisa and Florence as target domains. The longer is the value bar, the higher is the contribution of the corresponding feature. We focus on the top five values.

In general, we observe that there is not a clear pattern among the different classifiers and geographical units. Similarly to the observation reported in [6], for A1 in Figure 6, we have the presence of several IMN-based features like the betweenness of the movement from the first and third most important locations (l1l1_betweenness), the number of incoming edges in the second most visited location (l2_indegree), the events at the most important locations (tot_events_loc1), and the acceleration for reaching them (avg_max_acc_loc1). Moving the observations to Figure 7, we notice how all the classifiers highly rely on features related to events. This means that, when aggregating data from different sources, it becomes fundamental to predict a crash to discriminate along dimensions involving harsh accelerations, harsh braking, and harsh cornering. In particular, besides the events happening in general (like tot_duration_Q that means the total duration of harsh cornering), we notice how the focus is on events happening when driving towards the second most visited location (like tot_events_type_Q_loc2 that counts the number of harsh cornerings for going to loc2). Finally, we underline again how IMN-based features are important. For instance, with NN over Florence using A3 (bottom right of Figure 7) we have that the most important feature for deriving a car crash is avg_reg_mov_duration, i.e., the average duration of the movements performed regularly. This suggests that performing general actions to reduce the travel time for such a specific portion of the mobility can have a significant impact on the probability of a crash in the area, improving safety overall.

6 Conclusion and Future Work

In this paper, we have introduced the long-term car crash prediction problem, its associated task of risk assessment and the geographically transferred car crash prediction problem. For the first problem, we proposed a solution consisting in extracting sophisticated features of the user's mobility, able to

capture not only basic characteristics of her mobility, but also higher-level information derived from a network view of her mobility history as well as contextual knowledge directly inferred through analysis of the collective data of all users. On top of such features, machine learning models can be trained and successfully employed. Experiments on real data showed that our solution outperforms basic solutions based on state-of-art features, and a preliminary inspection of the prediction models through explainable AI methods allowed us to identify a few representative features associated with crash risk. For the second problem, the solution proposed consists in exploiting city indicators that can be derived from mobility data to design geographical transfer learning solutions based on the ensemble principle and weighted through city similarities. The experimentation on real data demonstrated that solutions employing city indicators for driving the transfer overcome standard baselines that do not use them. Explanation techniques also revealed some of the features that are most important for the success of the transfer learning methodology.

The results and insights obtained with this work opened several research and practical questions that we would like to address in the future, among which we mention the following. First, the IMN representation adopted in the driving modeling phase appears to be the right tool for enriching the data with higher-level semantics, such as the purpose of trips and stops, as done in [7], the driving *moods* (e.g., through unsupervised analysis of speeds and accelerations, or driving through dangerous intersections [11]), or by better describing the evolution of driving habits. Also, contextual data might be expanded, integrating several external, public data sources, such as the presence of Points of Interest, the road network structure, weather conditions, etc. While the model explanation tools were used in this work as a means for understanding the causes of crashes, their application can be further extended to improve the performance of the models by integrating feedback from domain experts – a *human in the loop* approach that can be made possible by model explanation itself. The city indicators we adopted, which are at the basis of our transfer learning proposals, are just a subset of a large spectrum of possible choices, our current purpose being to yield a general characterization of the urban areas involved. However, searching the optimal set of city indicators to reach the best model transferrability on the specific prediction problem would be indeed an interesting extension of the current work. Finally, geographical transfer learning is a poorly explored area, and the results discussed in this paper represent only a first step in this direction. More sophisticated solutions could be obtained by an appropriate combination of standard techniques (for instance, domain resampling for aligning distributions) and context-aware methods (e.g., the city indicators themselves or external information about the territory).

Acknowledgments. This work is partially supported by the European Community H2020 programme under the funding scheme *Track &Know* (Big Data for Mobility Tracking Knowledge Extraction in Urban Areas), G.A. 780754, https://trackandknowproject.eu/; and *SoBigData++*, G.A. 871042, http://www.sobigdata.eu.

Funding

This work is partially supported by the European Community H2020 programme under the funding scheme *Track &Know* (Big Data for Mobility Tracking Knowledge Extraction in Urban Areas), G.A. 780754, https://trackandknowproject.eu/; and *SoBigData++*, G.A. 871042, http://www.sobigdata.eu.

Conflicts of Interest

Not applicable

Ethics Approval

Not applicable

Consent to Participate

Not applicable

Consent for Publication

Not applicable

Availability of Data and Material

The vehicles datasets adopted in this work are private, and were provided within the scope of the Track & Know project (https://trackandknowproject.eu/), while the city indicators can be requested through the project web site.

Code Availability

The code is open source, and can be downloaded at: https://github.com/riccotti/CrashPrediction

Authors' Contributions

Mirco Nanni: Conceptualization, Methodology, Formal analysis, Investigation, Resources, Writing, Supervision, Project administration, Funding acquisition. Riccardo Guidotti: Conceptualization, Methodology, Formal analysis, Investigation, Software, Writing, Supervision. Agnese Bonavita: Methodology, Software, Validation, Investigation, Writing, Visualization. Omid Isfahani Alamdari: Methodology, Software, Validation, Investigation, Writing, Visualization.

References

[1] Longhi, L., Nanni, M.: Car telematics big data analytics for insurance and innovative mobility services. Journal of Ambient Intelligence and Humanized Computing, 1–11 (2019)

[2] Wang, Y., Xu, W., Zhang, Y., Qin, Y., Zhang, W., Wu, X.: Machine learning methods for driving risk prediction. In: Proceedings of the 3rd ACM SIGSPATIAL Workshop on Emergency Management Using, p. 10 (2017). ACM

[3] Lee, C., Hellinga, B., Saccomanno, F.: Real-time crash prediction model for application to crash prevention in freeway traffic. Transportation Research Record **1840**(1), 67–77 (2003)

[4] Ba, Y., *et al.*: Crash prediction with behavioral and physiological features for advanced vehicle collision avoidance system. Transportation Research Part C: Emerging Technologies **74**, 22–33 (2017)

[5] Cruz, L.A., *et al.*: Trajectory prediction from a mass of sparse and missing external sensor data. In: 2019 20th IEEE International Conference on Mobile Data Management (MDM), pp. 310–319 (2019). IEEE

[6] Guidotti, R., Nanni, M.: Crash prediction and risk assessment with individual mobility networks. In: 2020 21st IEEE International Conference on Mobile Data Management (MDM), pp. 89–98 (2020). IEEE

[7] Rinzivillo, S., *et al.*: The purpose of motion: Learning activities from individual mobility networks. In: 2014 International Conference on Data Science and Advanced Analytics (DSAA), pp. 312–318 (2014). IEEE

[8] Guidotti, R., *et al.*: There's a path for everyone: A data-driven personal model reproducing mobility agendas. In: 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 303–312 (2017). IEEE

[9] Nanni, M., Bonavita, A., Guidotti, R.: City indicators for mobility data mining. In: Big Mobility Data Analytics (BMDA) (2021). CEUR

[10] Wang, J., Xu, W., Gong, Y.: Real-time driving danger level prediction. Google Patents. US Patent 7,839,292 (2010)

[11] Salim, F.D., Loke, S.W., Rakotonirainy, A., Srinivasan, B., Krishnaswamy, S.: Collision pattern modeling and real-time collision detection at road intersections. In: 2007 IEEE Intelligent Transportation Systems Conference, pp. 161–166 (2007). IEEE

[12] Abdel-Aty, M.A., Pemmanaboina, R.: Calibrating a real-time traffic crash-prediction model using archived weather and its traffic data. IEEE Transactions on Intelligent Transportation Systems **7**(2), 167–174 (2006)

[13] Mannering, F.L., Bhat, C.R.: Analytic methods in accident research: Methodological frontier and future directions. Analytic methods in accident research **1**, 1–22 (2014)

[14] Kweon, Y.-J., *et al.*: Development of crash prediction models with individual vehicular data. Transportation research part C: emerging technologies **19**(6), 1353–1363 (2011)

[15] Lord, D., Mannering, F.: The statistical analysis of crash-frequency data: a review and assessment of methodological alternatives. Transportation research part A: policy and practice **44**(5), 291–305 (2010)

[16] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2009)

[17] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE **PP**, 1–34 (2020). https://doi.org/10.1109/JPROC.2020.3004555

[18] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)

[19] Bazzi, H., Ienco, D., Baghdadi, N., Zribi, M., Demarez, V.: Distilling before refine: Spatio-temporal transfer learning for mapping irrigated areas using Sentinel-1 time series. IEEE Geoscience and Remote Sensing Letters **17**(11), 1909–1913 (2020). https://doi.org/10.1109/LGRS.2019.2960625

[20] Syrris, V., Pesek, O., Soille, P.: Satimnet: Structured and harmonised training data for enhanced satellite imagery classification. Remote Sensing **12**, 3358 (2020). https://doi.org/10.3390/rs12203358

[21] Bappee, F.K., Soares, A., Petry, L.M., Matwin, S.: Examining the impact of cross-domain learning on crime prediction. J. Big Data **8**(1), 96 (2021). https://doi.org/10.1186/s40537-021-00489-9

[22] Liu, Z., Shen, Y., Zhu, Y.: Where will dockless shared bikes be stacked? — parking hotspots detection in a new city. In: Proc. of the 24th ACM SIGKDD. KDD '18, pp. 566–575. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3219819.3219920

[23] Iddianozie, C., McArdle, G.: A transfer learning paradigm for spatial networks. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied

Computing. SAC '19, pp. 659–666. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3297280.3297342

[24] Rogerson, P.A.: Statistical Methods for Geography: A Student's Guide. SAGE Publications, New York (2010). https://books.google.ch/books?id=Zz69Ab8i0QsC

[25] De Sherbinin, H.G. A.; Bittar: The role of sustainability indicators as a tool for assessing territorial. Environmental Competitiveness; International Forum for Rural Development (London, UK, 2003)

[26] Nélson, A., *et al.*: A comparative evaluation of mobility conditions in selected cities of the five brazilian regions. Transport Policy **37**, 147–156 (2015). https://doi.org/10.1016/j.tranpol.2014.10.017

[27] Gillis, D., Semanjski, I., Lauwers, D.: How to monitor sustainable mobility in cities? literature review in the frame of creating a set of sustainable mobility indicators. Sustainability **8**, 29 (2015)

[28] CITEAIR consortium: Air Quality in Europe web site. [Online; accessed 21-December-2020] (2007). http://www.airqualitynow.eu/

[29] Tafidis, P., *et al.*: Sustainable urban mobility indicators: policy versus practice in the case of greek cities. Transportation Research Procedia **24**, 304–312 (2017). https://doi.org/10.1016/j.trpro.2017.05.122. CSUM 2016, 26 – 27 May 2016, Volos, Greece

[30] Giannotti, F., *et al.*: Unveiling the complexity of human mobility by querying and mining massive trajectory data. The VLDB Journal **20**(5), 695–719 (2011)

[31] F., L., G., A., et al, A.N.: Citywide traffic analysis based on the combination of visual and analytic approaches. J geovis spat anal **4**(15) (2020)

[32] Trasarti, R., *et al.*: Mining mobility user profiles for car pooling. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1190–1198 (2011). ACM

[33] Guidotti, R., Trasarti, R., Nanni, M.: Tosca: two-steps clustering algorithm for personal locations detection. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 38 (2015). ACM

[34] Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), 379–423 (1948)

[35] Moran, P.A.P.: Notes on continuous stochastic phenomena. Biometrika **37**(1/2), 17–23 (1950)

[36] Saberi, M., Mahmassani, H.S., Brockmann, D., Hosseini, A.: A complex network perspective for characterizing urban travel demand patterns: graph theoretical analysis of large-scale origin–destination demand networks. Transportation **44**(6), 1383–1402 (2017)

[37] Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **2008**(10), 10008 (2008)

[38] Alonso, W.: A theory of movements: Introduction. Working Paper 266 (1976)

[39] Simini, F., Gonzalez, M.C., Maritan, A., Barabasi, A.-L.: A universal model for mobility and migration patterns. Nature **484**(7392), 96–100 (2012)

[40] Masucci, A.P., Serras, J., Johansson, A., Batty, M.: Gravity versus radiation models: On the importance of scale and heterogeneity in commuting flows. Physical Review E **88**(2), 022812 (2013)

[41] Porta, S., Crucitti, P., Latora, V.: Centrality measures in spatial networks of urban streets. Physical Review E **73**(3, part 2), 036125–1 (2006)

[42] Tan, P.-N., et al.: Introduction to data mining. Boston: Pearson Addison Wesley (2005)

[43] Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp. 4765–4774 (2017)

[44] Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research **7**(85), 2399–2434 (2006)

[45] Chakravarti, Laha, (1967), R.: Handbook of Methods of Applied Statistics, Volume I. John Wiley and Sons, Hoboken (1967)

[46] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)

[47] Tan, P.-N.: Introduction to Data Mining. Pearson Education India, Tamil Nadu (2018)