MD2SL SECOND LEVEL MASTER DEGREE IN DATA SCIENCE AND STATISTICAL LEARNING

IMT SCHOOL FOR ADVANCED STUDIES OF LUCCA, UNIVERSITY OF FLORENCE

GENERATION OF MOBILE PHONE TRAFFIC DATA

Supervisor
Professor Fabio Pinelli

Author
Alessandro Lo Verde
MATRICOLA: 7130258

Academic Year 2022-2023

Contents

Abstract

This study focuses on generating synthetic mobile traffic data using various features derived from areal data in the cities of Paris, Lyon, and Marseille. Relevant tasks to achieve this goal are the selection of relevant features and the exploration of the relationship between urban areas and mobile data traffic data. Machine learning models are implemented to generate the time series of traffic data and feature importance methods are implemented to understand which features are most relevant. The proposed analysis holds potential for various applications, such as urban planning, infrastructure optimization, and simulating the impact of policy and infrastructure changes on mobile data usage in urban environments.

Introduction

Based on the NetMob2023 challenge and dataset [3], the research questions addressed in this thesis are: Is it possible to generate a synthetic time series of mobile traffic data consumption for a specific area of a city, given certain geographic and demographic features and the possibility to train on actual traffic data available for other areas of the same city? Which features are relevant for this purpose? Furthermore, can the model trained in one city be extended to predict the time series of mobile traffic data in another city? Finally, are all the categories of mobile applications (social networks, gaming,..) for which the time series of mobile traffic data is available similarly predictable?

The different methods and models proposed aim at generating synthetic time series of mobile traffic data by exploiting the diversity between urban areas.

During the training and validation phase, it's essential to have the real time series of mobile traffic consumption for some areas. However, after training, the goal is to generate the time series based only on readily available demographic and geographic factors.

The preliminary step is to find a division of a city into a set of cells and then to select a set of features that can better capture the relationship between these urban areas and their respective level of mobile traf-

fic data consumption such as cell size, demographic indicators, points of interest, presence of infrastructure and services, distance from the city centre and other relevant information. It is reasonable to assume that different areas will have unique mobile traffic data patterns, with certain categories of mobile applications being used more than others. For example, areas with predominantly office buildings may have higher traffic patterns for work-related applications, while residential areas may have higher traffic for entertainment and leisure applications, particularly in the evening hours.

In Chapter 1, an exploratory data analysis is performed on the NetMob 2023 Challenge dataset [3], where the target variable of traffic data is presented as a time series from 19 March 2019 to 31 May 2019, collected for Paris, Lyon and Marseille. The analysis confirms that different areas, in terms of size and other features, have different levels of mobile data consumption. A KNN analysis supports the idea that there is a relationship between available features and mobile traffic data consumption in cities.

Chapter 2 introduces the ML methods for obtaining the time series and reducing the number of features to a smaller set of important features.

Chapter 3 presents the results of applying the models to the dataset. First, the models trained on Paris are used to generate the time series for Paris. Then, the models trained on the Paris data are used to synthesise traffic patterns for other cities: Lyon, Marseille. Finally, the model is trained and evaluated on a smaller set of features selected according to the feature importance methods described in chapter 2.

Chapter 1

Problem Definition and Dataset Description

In the following chapter, the problem to be tackled is formalized, then a brief description of the the target and of the features selected for the scope is provided and their relation assessed with a KNN analysis.

1.1 Problem Definition

A city is partitioned into a set of cells, where each cell represents a specific geographical area and is associated with a set of features that describe its characteristics, e.g., population density, road connectivity, land use, POI, satellite images, distance from the city centre and more.

Moreover for each cell, several historical time series are available, providing traffic volume data for different distinct groups or categories of similar mobile services (e.g., work, leisure, gaming, and so on). The goal is to implement a set of models – one for each category of mobile services – that take as input the set of features describing each cell and generate

the relative time series of traffic volume.

More formally, let:

- C be the set of K cells in which a city is divided, denoted by $C = \{Cell_1, Cell_2, ..., Cell_K\}$.
- F be the set of M features describing each cell, denoted by $F = \{Feature_1, Feature_2, ..., Feature_M\}$.
- For each category g of mobile services, let T be the historical time series of traffic volume, endowed with a certain temporal aggregation (e.g.hourly), represented as a vector of this kind $T^g = \{T_1^g, T_2^g, ..., T_L^g\}$, where L is the number of time points in the historical data.

The problem is then to find a set of models $M_1, M_2, ..., M_N$, one for each category, where $M^g : F \rightarrow T^g$ is the model for group g that maps the set of features of each cell to the corresponding time series of traffic volume.

Each model M aims to learn the underlying patterns and relationships between the features of the cells and the traffic volume for the g-th group of mobile services to output realistic and accurate synthetic time series data; in other words, the output should exhibit the temporal dynamics and fluctuations of the traffic volume observed in real data.

1.2 X - The Features of the Models

The features regarding points of interest of the cities are from OpenStreetMap[1], an open-source platform that creates and distributes free geographic data and mapping to the public, while the features regarding

[1] https://wiki.openstreetmap.org/wiki/Map_features

the electric consumption, demographics and infrastructures are from the Institut National de la Statistique et des Études Économiques (INSEE)[2], which is the National Statistical Office of France and collects and publishes information about the French economy and people and conducts the periodic national census.

1.2.1 INSEE Data

In order to divide the French cities into cells, this work has used the a standard called IRIS2000 developed by INSEE, which divides the country into equal-sized units with approximately 2000 residents each. Towns with over 10,000 inhabitants and many with 5,000 to 10,000 inhabitants are divided into multiple IRIS units, totalling around 16,100 IRIS units in France, including 650 in overseas departments. Any town not divided into IRIS units is considered a single IRIS unit itself.

IRIS cells for Paris are 2635, 411 for Lyon and 336 for Marseille.

The information set linked to each IRIS cell (INSEE dataset) covers several indicators, from demographic information, building descriptors and working population indicators; a set of 1526 variables grouped on 19 classes for all the 49461 IRIS cells.

From this set, 171 features (many of which are different ranges for different categories of e.g. age, income etc...) have been selected from categories that were considered more representative to understand the social/urban characteristics of each IRIS cell, namely:

- **Demographic indicators on the residents** (e.g. age, gender, job, french nationality, income, standards of living, etc...)

[2]https://www.insee.fr/

- **Housing information** (e.g. number of different types of houses, m^2 and age of the houses,...)
- **Electricity Consumption** (e.g. electricity consumption for residences, for the tertiary sector, for industries etc...)

Some features have been discarded since they were not available in every city or because their range of values were overlapping with respect to other features (e.g. age ranges of the population).

Some IRIS cells have been removed if the Revenue feature (e.g. $6e\_decile\_euro$) was equal to 0. In these cases it seems that there were clearly missing values that could impair the predictions since every IRIS cell has some inhabitant which has with all probability a revenue.

1.2.2 OpenStreetMap

In OpenStreetMap, map features are entities such as Points of Interest (POIs), road segments, areas devoted to some specific use, and more. Each map feature contains a geometry attribute: a point, segment, polygon, or multipolygon. Additionally, each feature is associated with a variable set of attributes that depend on the specific characteristics of the feature being described. Each map feature is categorized by means of a tag, which is a (key, value) pair. The key represents the feature's macro-category (e.g. shops), while the value specifies a sub-category within that macro-category (e.g. restaurant). Map features were selected from OpenStreetMap based on macro-categories that were suppose to be more relevant for IRIS cell characterization. These macro-categories are: amenity, healthcare, highway, historic, land use, leisure, office, public transport, railway, shop, tourism. For each macro-category, the most prevalent sub-

categories were selected. Any remaining subtype is categorized under the other IRIS cell feature. As a final step, for each IRIS cell, the absolute frequency number of map items falling within each selected subcategory were counted and considered as the final features.

The total number of POI features initially selected is 156 and the macro-categories with some sub-category (feature) of example is provided below:

- **Amenities** (e.g. bench, drinking water, restaurant, cafe, bar, school, book, bank, post box, etc...)
- **Healthcare** (e.g. pharmacy, hospital, dentist, etc...)
- **Historic** (e.g. memorial, monuments, ruins, museum, gallery, etc...)
- **Land use** (e.g. forest, grass, residential, etc...)
- **Leisure** (e.g. garden, sports centre, park, etc...)
- **Office** (e.g. estate agent, insurance, financial, government, etc...)
- **Public transport** (e.g. platform, stop position, station, etc...)
- **Railway** (rail, switch, crossing, bus stop, subway entrance, etc...)
- **Shops** (e.g. clothes, hairdresser, bakery, butcher, etc...)
- **Tourism** (e.g. attraction, artwork, hotel, etc...)

To these features a simple spatial feature **'distance_from_the_centre'** was calculated and added. This has been defined as the geodesic distance of each IRIS cell from the centre of the city. This feature has been added since it can be hypothesized that centrality can be a relevant factor to determine the mobile traffic data.

1.3 y - The Time Series of Mobile Traffic Data

The time series are obtained from the network traffic data from the NETMOB 2023 challenge [3], specifically the focus is on the cities of Paris, Lyon, and Marseille.

The original time series spans from 16th March 2019 to 31st May 2019.

Each series depicts the network traffic for a specific mobile application within a 100m x 100m cell, recorded at 15-minute intervals.

The categories of applications selected are the following:

- **Social Media Apps**: Facebook, Instagram, LinkedIn, Pinterest, Snapchat, and Twitter.
- **Gaming Apps**: Clash of Clans, EA Games, Fortnite, PlayStation, and Pokémon GO.

Initially, the time series for each specific 100m x 100m cell has been associated with the IRIS cell it intersected the most. Within the IRIS cell, the time series associated have been aggregated (summed) to form a single time series for the IRIS cel considered. The result was a set of time series for each IRIS cell for each application considered.

Next, the time series contained in each IRIS cell have been aggregated by their app category. As mentioned before, the two categories considered are Social and gaming. The result was a set of time series for each IRIS cell for both the categories considered.

Finally various temporal aggregations were applied to each time series:

- **Entire time series divided per 8 hours ranges** (night(0:00 - 8:00), morning(8:00 - 16:00), afternoon / evening (16:00-24:00): each time

series is aggregated (summed) in intervals of 8 hours. Every day is represented as 3 data points. It can be appreciated from the Figure 1.1 below how the average consumption of mobile data per iris cell is of two order of magnitude higher for social networks than for gaming in Paris. The consumption decreases during time from March to May (perhaps a better climate during May decreases smartphones use). Focusing on the single day pattern, the consumption of mobile traffic data has a minimum during the night hours, a higher use in the morning and a peak during the afternoon / evening hours.

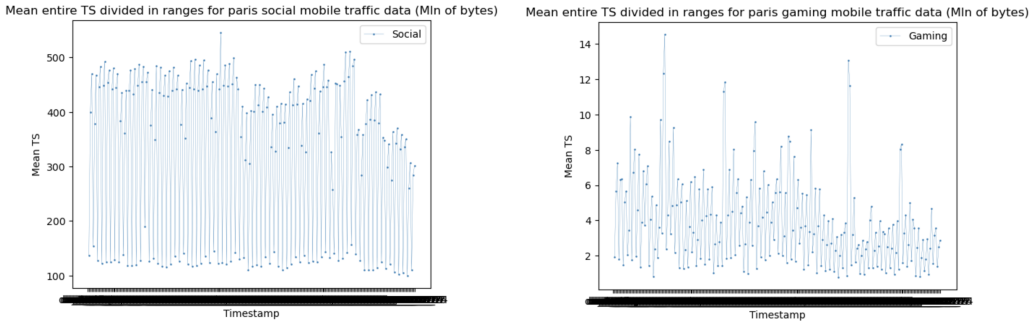

Figure 1.1: Entire series average mobile data consumption for Social in Paris divided into morning, afternoon and night ranges - social (left) gaming (right)

- **Weekly hourly average**: the time series aggregated hourly is averaged across all the weeks of the entire time series in order to have an averaged weekly hourly time series. In the picture below (Figure 1.2) it can be appreciated the average across all IRIS cells of the weekly hourly average time series of traffic data for social and gaming for the city of Paris. Social mobile traffic data is less busy on weekends, while gaming traffic remains steady daily, peaking on Wednesdays.

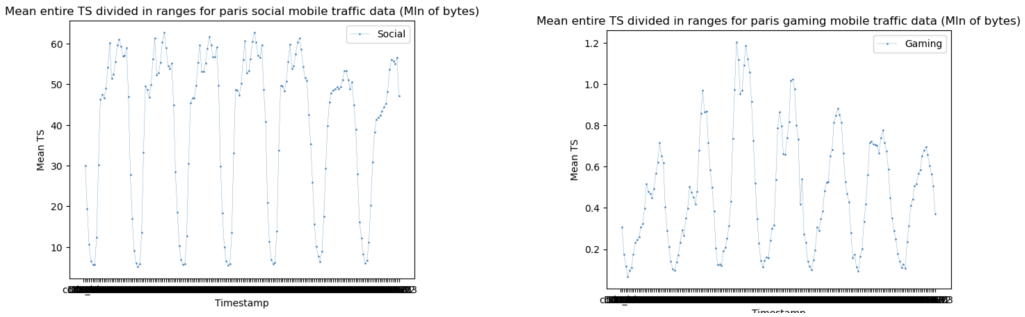

Figure 1.2: Weekly hourly average mobile data consumption for Social in Paris - social (left) gaming (right)

Trends and patterns of the traffic data are similar for the other cities even if the average data consumption per each area is smaller for Lyon and even smaller for Marseille (Figure 1.3). This factor can cause bias in the predictions when training on Paris and evaluating on the other cities, if this difference cannot be explained completely by the features selected, but it is related to something not included in the models (tourist flows, consumers preferences, etc...). This topic will be discussed in the conclusions.

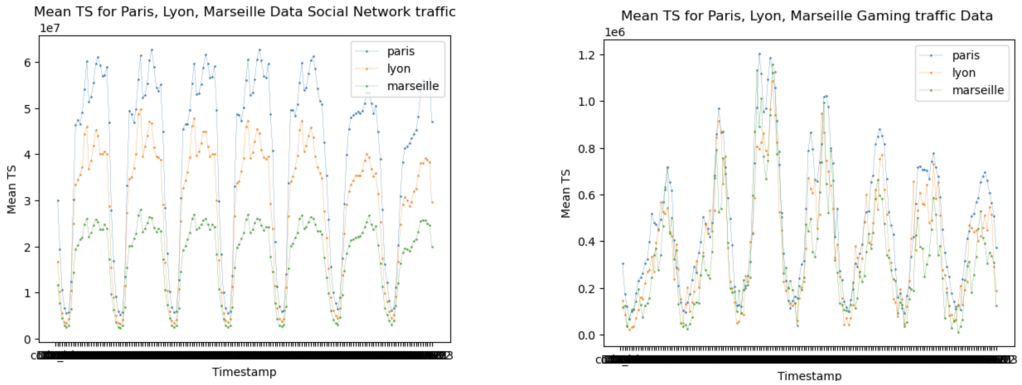

Figure 1.3: Weekly hourly average mobile data consumption across the different cities - social (left) gaming (right)

1.4 Relevance of the Selected Features with KNN

In the heat maps in Figure 1.4 and Figure 1.5 are presented the average data consumption and data consumption density (traffic data in the iris cell divided by its area) for social networks in Paris, Lyon and Marseille. It seems evident how the total data consumption is not independent from the size of the IRIS cell (the largest the cell, the higher the traffic data) and the data consumption density seems to be negatively correlated with the geodesic distance from the city centre (the closer to the city centre, the higher the traffic data). Perhaps other features (like the ones mentioned in the section before) characterizing the central and the peripheral cells can help in explain the difference in the level of traffic data consumed across the cells, but also the dynamics during time of their use.

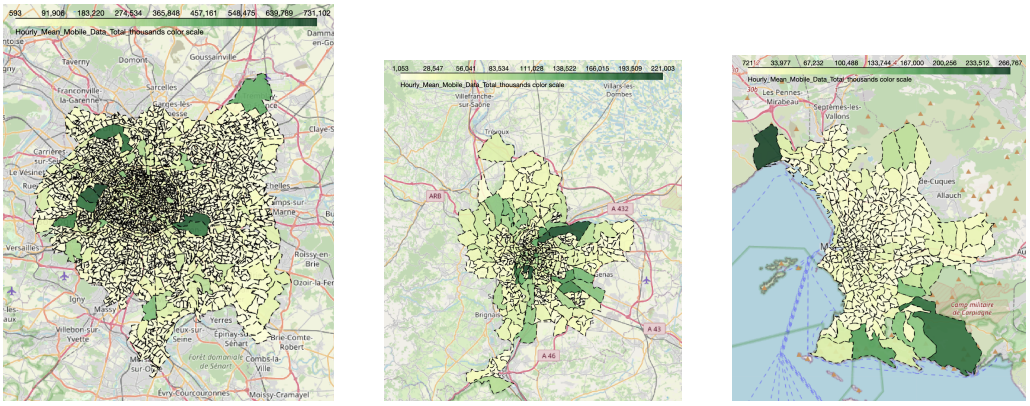

Figure 1.4: Distribution of the average traffic data used for Social (thousands of bytes) for Paris (left), Lyon (central), Marseille (right)

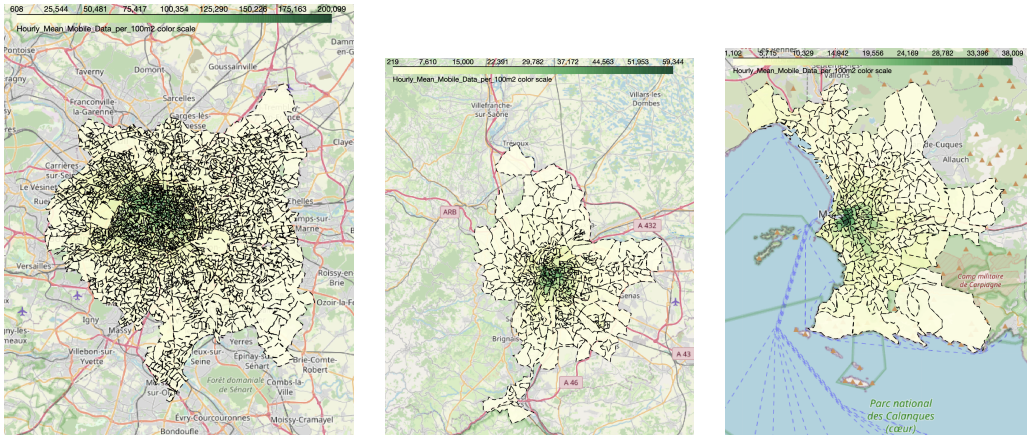

Figure 1.5: Distribution of the average traffic data density used for Social (bytes per 100 m2) for Paris (left), Lyon (central), Marseille (right)

The relevance of the features selected to predict the mobile traffic data will be analyzed in the next section.

To investigate and quantify the relationship between the X features and the target (time series of traffic data consumed per IRIS cell), a KNN method is employed. This method clusters a set of n similar IRIS cells (based on their X features) to a specific target IRIS cell. Subsequently, the time series of these clustered cells is evaluated by computing the Root Mean Square Error (RMSE) against the target cell's time series. Additionally, these results are compared with those obtained from randomly selecting n IRIS cells.

In the provided example (Figure 1.6) featuring the IRIS cell n: 751030601 situated in the centre of Paris, it's evident that the social data consumption average weekly time series associated with the 10 KNN neighbors of the selected target IRIS cell closely aligns with the target, resulting in an average smaller RMSE compared to the one computed on the time series of the randomly selected cells.

The average RMSE calculated across all the IRIS cells of Paris (2635

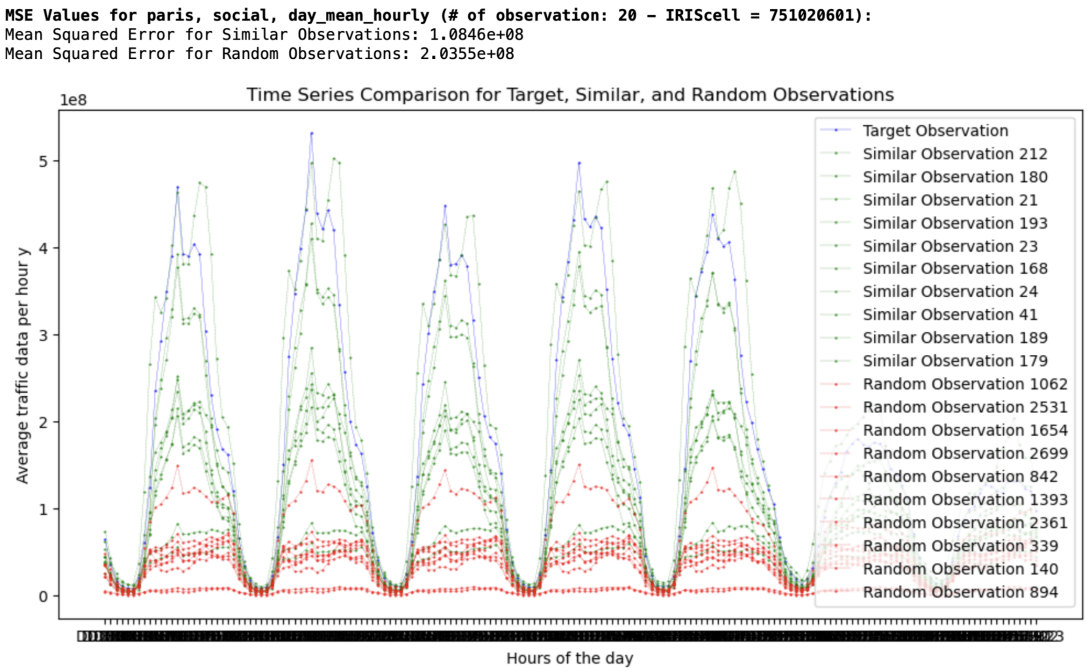

Figure 1.6: KNN on 10 neighbours, Target IRIS cell n: 751030601

cells), Lyon (411 cells) and Marseille (336 cells) are summarized in the table in the next page. A different number of n similar and n random neighbours IRIS cells are taken into consideration for the calculations. A smaller mean RMSE is evident when the time series are clustered based on the X features with respect to a cluster of random time series.

City	Category	Neighbours (%)		Overall Mean RMSE	Overall Mean RMSE
		Percentage	# cells	Similar Neighbours	Random Neighbours
Paris	Social	0.01	26	5.46×10^7	7.81×10^7
	Gaming	0.01	26	5.86×10^5	8.98×10^5
	Social	0.05	131	6.32×10^7	7.89×10^7
	Gaming	0.05	131	6.35×10^5	8.93×10^5
	Social	0.1	263	6.74×10^7	7.87×10^7
	Gaming	0.1	263	6.53×10^5	8.98×10^5
Lyon	Social	0.01	4	7.90×10^6	2.89×10^7
	Gaming	0.01	4	5.66×10^5	7.95×10^5
	Social	0.05	20	9.66×10^6	2.19×10^7
	Gaming	0.05	20	6.03×10^5	7.75×10^5
	Social	0.1	41	9.45×10^6	2.26×10^7
	Gaming	0.1	41	6.10×10^5	7.46×10^5
Marseille	Social	0.01	3	7.14×10^6	1.39×10^7
	Gaming	0.01	3	3.03×10^5	5.70×10^5
	Social	0.05	16	9.77×10^6	1.20×10^7
	Gaming	0.05	16	3.06×10^5	5.45×10^5
	Social	0.1	33	1.07×10^7	1.24×10^7
	Gaming	0.1	33	3.36×10^5	5.78×10^5

Table 1.1: Summary of the average RMSE calculated over all IRIS cells for different combinations of city, category and n nearest neighbours. For each target cell, the RMSE between its time series and a cluster of n similar neighbours (KNN) or n random neighbours has been calculated.

Chapter 2

Models and Methods

2.1 Models Definition

The experiments contained in chapter 3 focus on evaluating the performance of the models and methods discussed in this section.

2.1.1 Neural Network model

The first model chosen is a fully connected neural network with multiple layers. For a comprehensive introduction on these class of models these two references are provided in [6], [7].

This class of models has been chosen for its ability to capture complex relationships between input features and multiple output variables (in this case the time points of the time series).

Different standard architectures (number of neurons, layers, level of dropout) have been tested in order to get satisfying results without increasing too much the computational times. A satisfying trade-off architecture selected for this work is the following one:

- **3 Hidden layers with 2000 neurons each**: Each hidden layer employed Rectified Linear Unit (ReLU) activation functions to introduce non linearity into the network, L1LOSS loss function (mean absolute error) and Adam optimizer. The input dimension corresponds to the number of features in the dataframe (328).

- **Output layer**: Positioned at the network's end, this layer is responsible for generating predictions on the time series of mobile data consumption for each cell. Its dimension corresponds to the number of time points of the time series aggregation chosen (168 for the weekly hourly average and 228 for the entire series divided in 8 hr ranges).

- **Dropout: 0.2**. A dropout of 0.2 was inserted between consecutive hidden layers. Dropout deactivates 20 % of the neurons randomly during each forward pass during training to mitigate overfitting.

- **Number of epochs: 150/200**. The number of epochs is around 150/200 epochs for each combination of time aggregation and category selected.

- **learning rate**: The optimal learning rate was determined depending on the time aggregation method and category selected. The tuning of the learning parameter has been achieved through a cross-validation process or by analyzing training and validation error graphs over epochs to monitor convergence and detect signs of overfitting or underfitting.

  The size of the batch (number of IRIS cells passed to the model at each iteration during the training phase) is 100. A smaller batch size

decreased performance, while a larger batch size didn't allowed a smooth decrease of the validation error during training.

2.1.2 XGBOOST

The second model chosen for the analysis is XGBoost, defined by its authors as a "scalable end-to-end tree boosting system", known for its efficiency and good performances on different tasks. It was implemented with multi output (since the target are the data points of the time series to be predicted).

All the model parameters that could be tuned (with the exception of the learning rate (eta) that is the only parameter which changes with the category and time aggregation of the problem) have been tuned according to a 3 folds cross-validation on the Social weekly hourly average time series and then applied to all other categories and time aggregations.

Instead of optimizing all parameters at once, the fine tuning process examined 3-4 alternatives for each parameter in pairs, reducing the computational load. This pairwise approach balanced efficiency with feasibility, allowing a simple laptop to manage the task without the high demands of a full grid search, although it sacrificed some precision. Finally, if no significant improvement was observed between two or more parameter combinations, the selection leaned towards the most conservative option in terms of computation time.

Better results can be achieved by tuning all the parameters according to the specific characteristics of each type of time series.

The specifics of the model configuration are as follows:

- **tree_method: "hist"**. It indicates that a histogram-based algorithm

is utilized for tree construction. This method creates histograms for each feature to find the best splits. It's faster and more scalable than "exact", making it suitable for large datasets.

- **objective: "reg:squarederror"**. The loss function to be minimized during training is the squared error between predictions and actual values, a common option for regression tasks.

- **booster: "gbtree"**. It specifies that a tree-based boosting algorithm will be employed, where the model is constructed by adding decision trees iteratively.

- **eval_metric: "rmse"**. The square root of the average squared differences between predicted and actual values is the evaluation metric used to assess model performance during training and validation.

- **multi_strategy: "multi_output_tree"**. Strategy to handle multi-output problems. Unlike traditional tree models, which provide a scalar value at each leaf, vector leaf tree models provide a vector at each leaf, allowing for a more complex representation of the data. This strategy was employed to deal with a target variable (y) with multiple columns.

- **enable_categorical: True**. A boolean indicating whether categorical features should be treated as such.

- **seed: 300**: The random seed used for the reproducibility of the results. Any randomization in the model (such as during the training of trees) produces the same outcomes when repeated, given that all other factors remain constant.

- **early_stopping_rounds: 7**. Number of rounds (boosting iterations) that can occur without improvement in the evaluation metric before stopping the training process. Here, set to 7, enabling early stopping to prevent overfitting. To prevent overfitting and to ensure efficient training, the training process will stop if the validation error does not improve for five consecutive boosting rounds.

- **n_estimators: 512/2000**. The number of boosting trees (or rounds) to be used in building the model. Initially set to 512 to control the complexity of the model when tuning the other parameters, it has then been set to 2000.

- **learning_rate or eta**: The step size shrinkage used in each boosting iteration. After each boosting step, eta shrinks the feature weights to make the boosting process more conservative. It is adapted depending on the time aggregation and category.

- **min_child_weight: 5**. It's the minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning. In linear regression task, this simply corresponds to minimum number of instances needed to be in each node. The larger min_child_weight is, the more conservative the algorithm will be.

- **max_depth: 5**. Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. It has been limited to 5 levels in order to control the complexity of individual trees.

- **subsample: 0.8**: Subsample ratio of the training instances. Setting it to 0.8 means that XGBoost would randomly sample the 80% of the training data prior to growing trees and this will prevent overfitting.
- **colsample_bytree: 0.8**. It's the subsample ratio of columns considered when constructing each tree.
- **gamma: 0**. Minimum loss reduction required to make a further partition on a leaf node. The larger gamma is, the more conservative the algorithm will be.
- **reg_alpha: 0**. L1 regularization term on weights. Set to 0, indicating no L1 regularization.
- **reg_lambda: 0.05**: L2 regularization term on weights.

2.2 Features Selection Methods

Machine learning models often operate in complex feature spaces, making it crucial to understand how each feature contributes to the predictions. While linear models offer straightforward methods for calculating individual feature effects, such approaches may not generalize to other model types. Here two approaches are taken into consideration and defined to compute feature importance in ML models.

In chapter 3 the most important 50 features will be selected based on the importance measure defined here and will be used to fit a reduced version of the previous models in order to understand which features are the most important, if features selection can help in prevent overfitting and if all the 328 features considered intially are necessary to predict mobile traffic data with precision.

2.2.1 Permutation Feature Importance

The permutation feature importance method is a technique commonly used in machine learning to evaluate the importance of each feature in a predictive model. Unlike Shapley values, which are based on cooperative game theory, permutation feature importance is an empirical method that directly assesses the impact of shuffling or permuting each feature on the model's performance.

Given a trained model f, typically a machine learning algorithm such as a decision tree or a neural network, and a dataset with feature matrix X and target variable (or vector) y, the permutation feature importance (PFI_i) for a feature i is calculated as follows:

$$\text{PFI}_i = \frac{1}{N} \sum_{j=1}^{N} \Big(\text{score}(X_i^{(j)}, y) - \text{score}(X_i, y) \Big)$$

where:

- N is the number of permutations. In this work it is set to 10.
- $X_i^{(j)}$ represents the dataset with the i-th feature randomly shuffled in the j-th permutation.
- $\text{score}(X_i^{(j)}, y)$ is the model's performance metric (MSE is used in this work) on the permuted dataset.
- $\text{score}(X_i, y)$ is the model's baseline performance metric (MSE is used in this work) on the original dataset.

This method assesses the change in model performance when the values of a particular feature are randomly shuffled while keeping other features intact. A significant decrease in performance (increase in the error score) using the dataset with the shuffled i-th feature indicates that

the i-th feature is particularly important for the model, as shuffling disrupts the model's ability to make accurate predictions. Conversely, if shuffling the i-th feature has little effect on performance, it suggests that the i-th feature may not be crucial for the model's predictions. The larger the value of the PFI_i, the most important is the i-th feature.

By sorting in descending order the PFI_i for each i feature, an ordered list of the most important features is obtained.

2.2.2 Shapley Values and Shap Feature Importance

The Shapley value is a concept borrowed from cooperative game theory which offers insights into individual feature contributions regardless of model type (model-agnostic).

The Shapley value for a certain observation represents the contribution of a feature value to the model's prediction payout, considering all possible combinations of feature values. Mathematically, it is defined as:

$$\phi_j(\text{val}) = \sum_{S \subseteq \{1,\dots,p\} \setminus \{j\}} \frac{|S|!(p-|S|-1)!}{p!} (\text{val}(S \cup \{j\}) - \text{val}(S))$$

Here, S denotes a subset of features, p is the total number of features, and val(S) represents the prediction for feature values in set S that are marginalized over features that are not included in set S:

$$\text{val}_x(S) = \int \hat{f}(x_1, \dots, x_p)\, dP_{x \notin S} - E_X(\hat{f}(X))$$

So, val(S) indicates the prediction when only the features in set S are used, while the other features are treated as if they are distributed randomly (in fact they will be shuffled in the MC method described below).

The Shapley value satisfies several key properties, ensuring fairness and interpretability:

- **Efficiency**: Feature contributions sum up to the difference between the prediction for a specific observation and the average prediction.
- **Symmetry**: Features with equal contributions across all possible coalitions have the same Shapley value.
- **Dummy**: Features that do not alter predictions have a Shapley value of 0.
- **Additivity**: Shapley values for combined payouts follow a straightforward additive rule.

To give an intuition of the meaning of this method, Molnar in [1] provides the following illustration: "Suppose the feature values enter a room in random order. All feature values in the room participate in the game (= contribute to the prediction). The Shapley value of a feature value is the average change in the prediction that the coalition already in the room receives when the feature value joins them". Another intuitive analogy describes the Shapley value as the average change in prediction when a feature value joins a coalition of already present feature values. This conceptualization helps grasp its significance in understanding feature contributions.

The Shapley value of a feature value is not the difference of the predicted value after removing the feature from the model training. The interpretation of the Shapley value is: given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction is the estimated Shapley value.

The computation of the Shapley value involves evaluating predictions for various coalitions of feature values. Exact computation of the Shapley value becomes often impractical for models with numerous features due to the exponential growth in possible coalitions. Strumbelj et al. (2014) [7] propose an approximation with Monte-Carlo sampling:

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} (\hat{f}(x_{+j}^m) - \hat{f}(x_{-j}^m))$$

Where:

- $\hat{f}(x_{+j}^m)$ is the prediction for x, but with a random number of feature values replaced by feature values from a random data point z, except for the respective value of feature j for which the Shapley value must be calculated.
- The x-vector x_{-j}^m is almost identical to x_{+j}^m but the value x_j^m is also taken from the sampled z.
- For features that appear left of the feature x_j, the values are taken from the original observations, and for the features on the right, the values are taken from a random instance.

The algorithm to calculate the Shapley value through Monte Carlo works as follows:

1. Select an observation of interest x, a feature j, and the number of iterations M.
2. For each iteration:
   (a) Select a random observation z from the data.
   (b) Generate a random order of the features.

(c) Create two new observations:

- Observation x_{+j} is the original observation of interest x combined with the sample z, such that all values in the order after feature j are replaced by feature values from sample z.
- Observation x_{-j} is the same as x_{+j}, but with feature j replaced by the value for feature j from sample z.

(d) Compute the difference in the prediction from the black box:

$$\phi_{mj} = \hat{f}(x^m_{+j}) - \hat{f}(x^m_{-j})$$

3. Average all these differences:

$$\phi_j(x) = \frac{1}{M}\sum_{m=1}^{M}\phi_j^m$$

Once Shapley values have been calculated for a certain observation (IRIS cell) then it is important to have a method to understand feature importance across all data observations (in this work across all the IRIS cells). This task is done with Shap feature importance.

The idea behind SHAP feature importance is simple: Features with large absolute Shapley values are important. So in order to get the global importance, the absolute Shapley values for each j-th feature are averaged across each i-th observation (IRIS cell):

$$I_j = \frac{1}{n}\sum_{i=1}^{n}\left|\phi_j^{(i)}\right|$$

Where:

- I_j represents the SHAP feature importance of feature j.

- n is the total number of observations (cells).
- $\phi_j^{(i)}$ is the Shapley value of feature j for the observation i.

Finally, the features are sorted by decreasing importance in order to have an ordered list of the most important features.

SHAP feature importance is an alternative to permutation feature importance. There is a big difference between the two importance measures: Permutation feature importance is based on the decrease in model performance. SHAP is based on the magnitude of feature contributions.

Chapter 3

Results from the Experiments

3.1 Results from NN and XGBOOST Models

The NN and Xgboost models presented in chapter 2 have been trained and validated in Paris, since Paris has the largest number of IRIS cells, providing a richer and more varied dataset for training and validation. The IRIS cells for each city were divided into three partitions - 70% for training, 10% for validation, and 20% for testing. The same cells splits within a city (seed = 300) were applied for all the different categories and time aggregations outlined in chapter 1.

The models trained and validated on Paris have been evaluated in the test set of Paris, but also on the test sets of the other two cities: Lyon, and Marseille.

Here is the list of possible combinations for types of model, categories, and time aggregation analyzed:

- Type of Model: NN/Xgboost models
- Category of the time series: Social/gaming

- Time aggregation of the time series: weekly hourly average/entire time series divided in 8 hrs ranges

- Complete model, vs reduced model: the complete model comprehend all the 328 features initially selected, while the reduced model is trained using only the top 50 features according to the PFI method for the NN model and the SHAP feature importance method for XGboost. Which are the most important features according to these methods is presented in the next section.

Evaluation of model performance has been conducted utilizing two distinct error metrics: mean squared error (MSE) and mean absolute percentage error (MAPE).

The $MAPE_t$ is calculated as the average percentage of absolute errors across all the IRIS cells between the predicted values and the observed values for a certain point in the time series to be predicted. The formula to calculate the $MAPE_t$ for a certain point in time t is as follows:

$$MAPE_t = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100\% \tag{3.1}$$

The $RMSE_t$ is calculated as the square root of the average across all IRIS cell of the squared differences between the predicted values and the corresponding true values of the traffic data for a certain point in time for the time series to be predicted. The formula to calculate the $RMSE_t$ for each point in time t is as follows:

$$RMSE_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{3.2}$$

Where:

- Y_i represents the observed value.
- $\hat{Y}_i$ represents the value predicted by the model in a certain cell and point in time t.
- n is the total number of IRIS cells in the set of evaluation.

The average MAPE (Mean Absolute Percentage Error) and RMSE (Root Mean Square Error) across all time points in the predicted time series serve as summary metrics to assess the performance of different models with a single value:

$$MAPE = \frac{1}{T}\sum_{t=1}^{T} MAPE_t \tag{3.3}$$

$$RMSE = \frac{1}{T}\sum_{t=1}^{T} RMSE_t \tag{3.4}$$

After reviewing the summary error Table 3.1 (see next page) and analysing various error plots, the main evidence is as follows.

Looking at the **predictions made by the model trained on Paris and evaluated on Paris**, it follows that

- The **type of model** seems to have a moderate influence on the results.
  - The NN models seem to perform better than the XGBOOST models for the social category (both RMSE and MAPE) for both the time aggregations.
  - The NN models seem to perform better than the XGBOOST models for the gaming category in terms of MAPE, but worse in terms of RMSE for both the time aggregations.

Table 3.1: Error Metrics for Different Model Configurations

City	Category	Time Aggregation	Metric	NN	PFI (NN)	XGBoost	Shap (XG)
Paris trained on Paris	Social	Weekly (1hr)	Av_TS	3.75×10^7	3.75×10^7	3.75×10^7	3.75×10^7
			RMSE	1.86×10^7	2.06×10^7	1.93×10^7	1.86×10^7
			MAPE	44.46%	49.68%	48.14%	46.42%
		Entire Series (8hr)	Av_TS	3.02×10^8	3.02×10^8	3.02×10^8	3.02×10^8
			RMSE	1.45×10^8	1.58×10^8	1.56×10^8	1.52×10^8
			MAPE	43.62%	47.07%	46.03%	46.96%
	Gaming	Weekly (1hr)	Av_TS	4.76×10^5	4.76×10^5	4.76×10^5	4.76×10^5
			RMSE	8.30×10^5	8.40×10^5	7.73×10^5	7.73×10^5
			MAPE	89.83%	89.69%	250.23%	268.59%
		Entire Series (8hr)	Av_TS	3.85×10^6	3.85×10^6	3.85×10^6	3.85×10^6
			RMSE	8.71×10^6	8.77×10^6	8.19×10^6	8.20×10^6
			MAPE	88.23%	84.66%	274.72%	278.65%
Lyon trained on Paris	Social	Weekly (1hr)	Av_TS	2.63×10^7	2.63×10^7	2.63×10^7	2.63×10^7
			RMSE	1.84×10^7	1.97×10^7	2.07×10^7	2.09×10^7
			MAPE	71.83%	102.39%	115.09%	110.13%
		Entire Series (8hr)	Av_TS	2.13×10^8	2.13×10^8	2.13×10^8	2.13×10^8
			RMSE	1.59×10^8	1.59×10^8	1.97×10^8	1.65×10^8
			MAPE	80.27%	101.03%	120.95%	116.37%
	Gaming	Weekly (1hr)	Av_TS	3.61×10^5	3.61×10^5	3.61×10^5	3.61×10^5
			RMSE	5.72×10^5	5.88×10^5	6.81×10^5	6.69×10^5
			MAPE	120.13%	151.67%	430.45%	451.13%
		Entire Series (8hr)	Av_TS	2.92×10^6	2.92×10^6	2.92×10^6	2.92×10^6
			RMSE	5.56×10^6	5.62×10^6	6.36×10^6	6.31×10^6
			MAPE	122.16%	161.38%	482.93%	491.92%
Marseille trained on Paris	Social	Weekly (1hr)	Av_TS	1.63×10^7	1.63×10^7	1.63×10^7	1.63×10^7
			RMSE	2.14×10^7	2.58×10^7	2.98×10^7	2.87×10^7
			MAPE	129.23%	168.32%	193.88%	180.79%
		Entire Series (8hr)	Av_TS	1.31×10^8	1.31×10^8	1.31×10^8	1.31×10^8
			RMSE	1.93×10^8	1.99×10^8	2.42×10^8	2.24×10^8
			MAPE	151.43%	167.12%	196.45%	194.57%
	Gaming	Weekly (1hr)	Av_TS	3.23×10^5	3.23×10^5	3.23×10^5	3.23×10^5
			RMSE	5.12×10^5	5.23×10^5	5.76×10^5	5.72×10^5
			MAPE	162.67%	224.99%	570.40%	618.67%
		Entire Series (8hr)	Av_TS	2.62×10^6	2.62×10^6	2.62×10^6	2.62×10^6
			RMSE	5.25×10^6	5.33×10^6	5.85×10^6	5.86×10^6
			MAPE	179.09%	253.20%	661.39%	680.46%

- The **category (Social, Gaming)** of the time series seems to be relevant for the performance of the predictions (regardless from the type of model and time aggregation chosen):
  - The Social time series seems to be the easiest to predict with a lower error (average MAPE between different models and time aggregations of 40-50%).
  - The Gaming time series seems to be less easy to predict with an average MAPE of 80-100% over the different combinations of time aggregations for the NN model. XGBOOST, which is mostly fine-tuned to the social time series, also fails to predict the gaming time series (higher MAPE compared to NN although it has a lower RMSE compared to NN).

  This difference in performance can be seen in the scatter plot of all predicted values by the NN model for the weekly time series against the actual values below (Figure 3.1). Similar plots are produced by the XGBOOST model and the models trained on the entire series time aggregation.

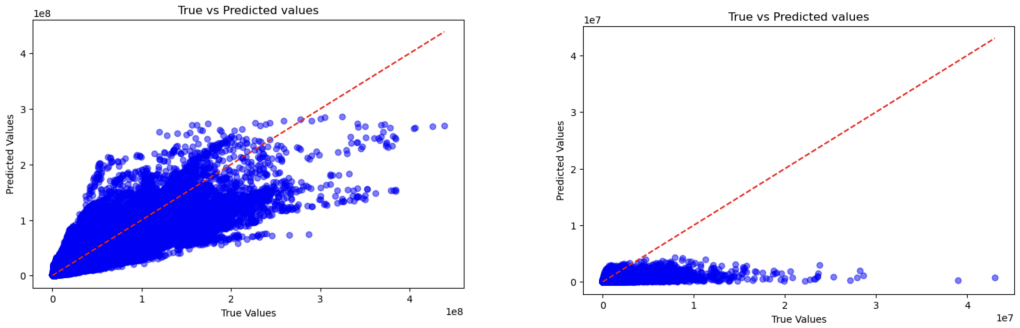

Figure 3.1: Scatter predicted vs true values of the weekly time series predicted with the NN model trained on Paris and evaluated on Paris - Social (left) Gaming (right)

- The **time aggregation** selected has little effect on the performance (similar MAPE):

  – If a model is good at predicting the average weekly consumption behaviour for a given cell, it will also be good at reproducing this behaviour for the same cell over the entire observation period, as can be seen in Figure 3.2.

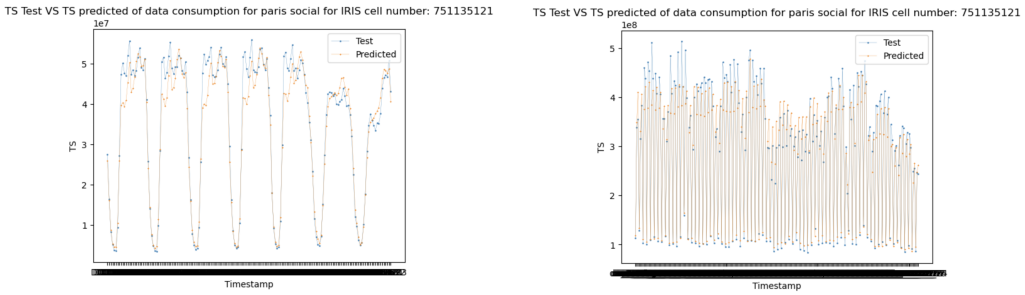

Figure 3.2: IRIS Cell: '751135121'. Social time series predicted by the NN model trained on Paris and evaluated on Paris - Weekly TS (left) Entire Series TS (right)

  – If a model strongly over- or under-predicts the average weekly traffic data, it is likely to show the same tendency when forecasting the whole time series. Even in these cases, the model can still capture the daily pattern with some accuracy. Figure 3.3 shows an example of overprediction for a certain IRIS cell.

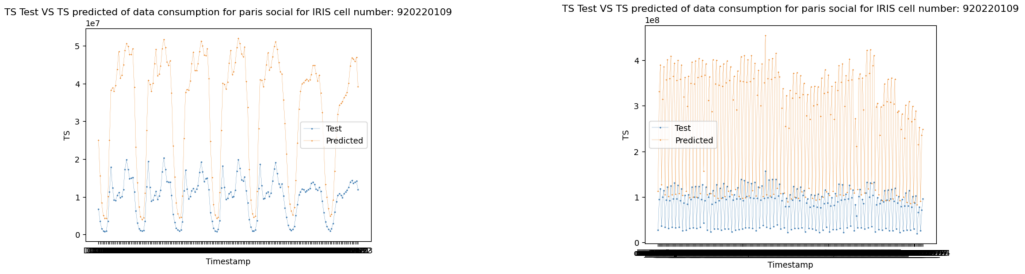

Figure 3.3: IRIS Cell: '920220109'. Social time series overpredicted by the NN model trained on Paris and evaluated on Paris - Weekly TS (left) Entire Series TS (right)

- Figure 3.4 shows the **geographical distribution in Paris of the MAPE errors** evaluated on the test set of IRIS cells for social (left) and gaming (right) using the NN model (similar behaviour can be observed for the XGBOOST model). It seems clear that:
  - While over-predictions (red and orange cells) are as frequent as under-predictions (in green), the magnitude of the over-predictions is often significantly greater than that of the under-predictions (as can also be seen from the colour bar range).
  - Over and under-predictions in the social and gaming error maps tend to be located in the same areas.
  - Iris cells with the same type of error within the same city seem to form distinct local clusters.

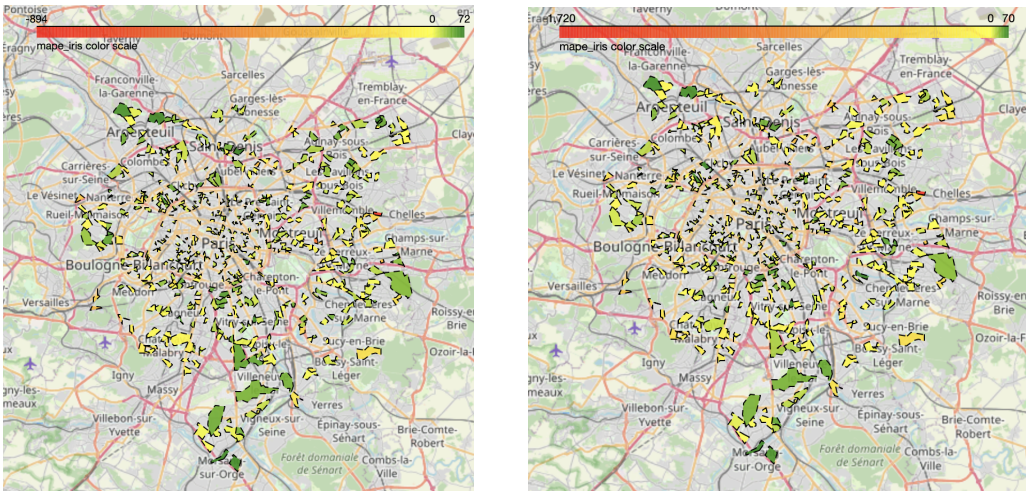

Figure 3.4: Distribution of the average MAPE for the weekly time series predicted with the NN model trained on Paris and evaluated on Paris for Social (left) and Gaming (right) - Over-pred. (red orange), good pred. (yellow), under-pred. (green)

- The **reduced models trained on the top 50 features according to the different measures of feature importance** give similar results

when compared to the complete ones:

  - The reduced NN models trained on the top 50 features ranked according to the Performance Feature Importance (PFI) method achieve basically the same results for the gaming time series and slightly worse results for the social time series.
  - The reduced XGboost models, trained on the top 50 features ranked according to the SHAP feature importance method, achieve essentially the same results as the complete models for all the time series combinations.

Looking at the **predictions made by the model trained on Paris and evaluated on Lyon and Marseille**, the evidence is as follows:

- The models trained on the social time series tend to predict better than those trained on the gaming time series, both in terms of RMSE and MAPE (for both time aggregations).
- The models tested on Lyon show an overall better performance compared to those tested on Marseille.
- Reduced models give similar results, sometimes better (XGBOOST with feature selected with SHAP), sometimes worse (NN with PFI).
- The time pattern during the different days and hours seems to be captured, but there is a **upward bias** in the predictions (especially in the social category), regardless of the type of model chosen (Figure 3.5). Figure 1.3 shows that the average traffic data consumption is highest in Paris, slightly lower in Lyon and even lower in Marseille. The features used in the models are not completely sufficient

to capture the differences between cities in average mobile traffic data. This point will be discussed further in the conclusions.

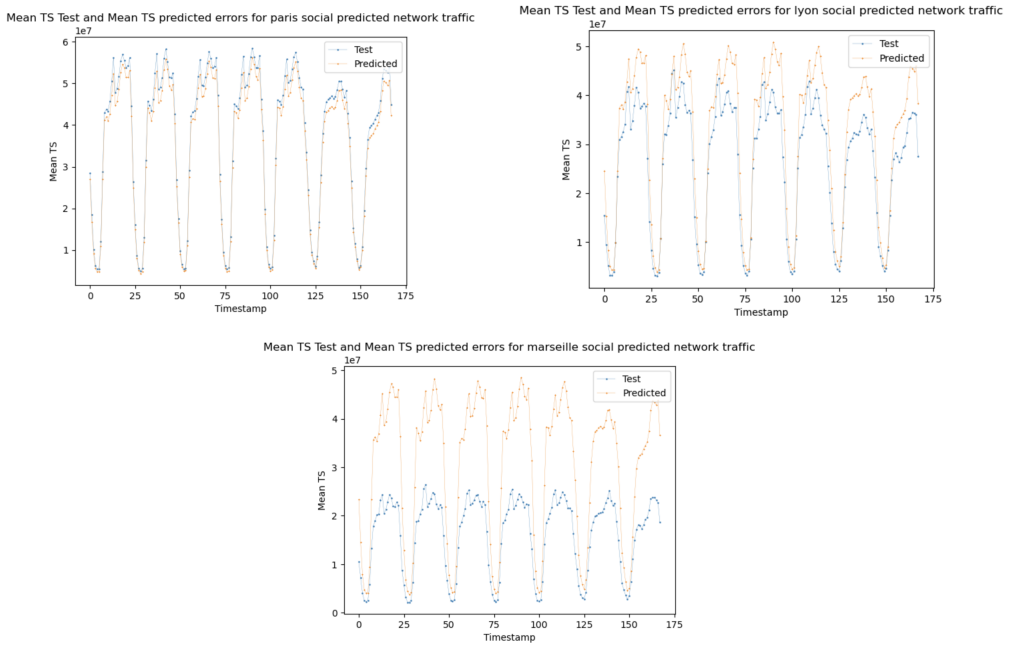

Figure 3.5: Weekly hourly TS average prediction across all IRIS cells vs true values of the average TS using the NN model trained on Paris and evaluated on: Paris (left), Lyon (right), Marseille (bottom)

3.2 Results from Features Importance Methods

The 50 most important features were selected on the basis of the the importance measure defined in Chapter 2, and are used to fit a reduced version of the previous models to understand which features are the most important, if feature selection can help to avoid overfitting and if all of the 328 features initially considered are necessary.

The Performance Feature Importance method was applied to the neural network models. The average PFI was calculated over 10 permuta-

tions for each feature column to obtain a consistent result. Shap feature importance was applied to Xgboost. Both methods were applied to all categories and time aggregations.

Figure 3.6 shows the **20 most important features using the Permutation Feature Importance (PFI) method applied to the Neural Network model** predicting the weekly time series for social and gaming data.

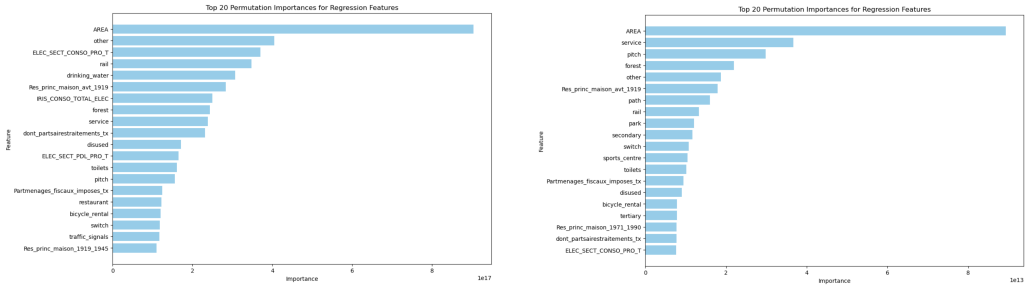

Figure 3.6: Top 20 features according to PFI - Social (left), Gaming (right)

Figure 3.7 illustrates the top 20 features by importance, as determined by the SHAP Feature Importance method applied to the XGBoost model predicting the weekly TS of Social and Gaming data. Similar results for PFI and SHAP importance can be obtained for the entire series TS.

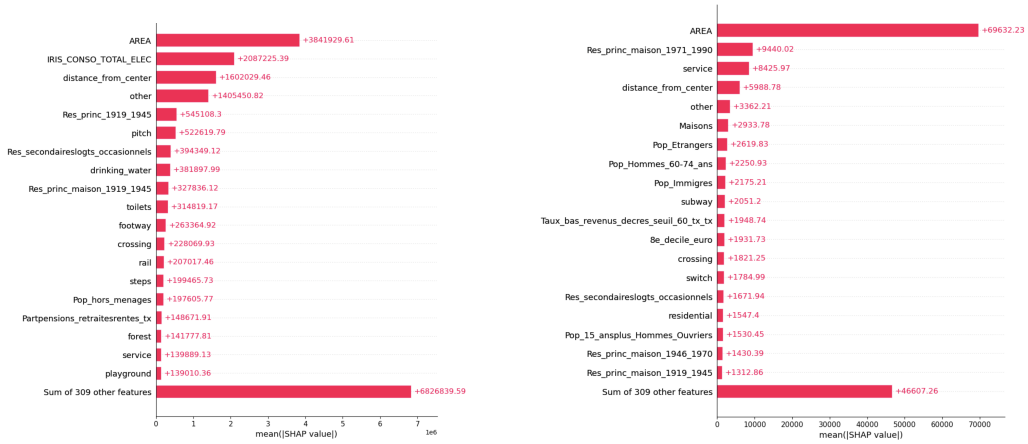

Figure 3.7: Top 20 features according to Shap - Social (left), Gaming (right)

The SHAP Summary plots (Figure 3.8) and dependence plots (not shown) gave an idea of the positive or negative contributions of these features. A summary plot combines feature importance and the sign of the contributions given by the features. Each point is a Shapley value for a specific feature and IRIS cell. The ordered features are on the y-axis, the x-axis is its Shapley value, and the color represents the feature's actual value.

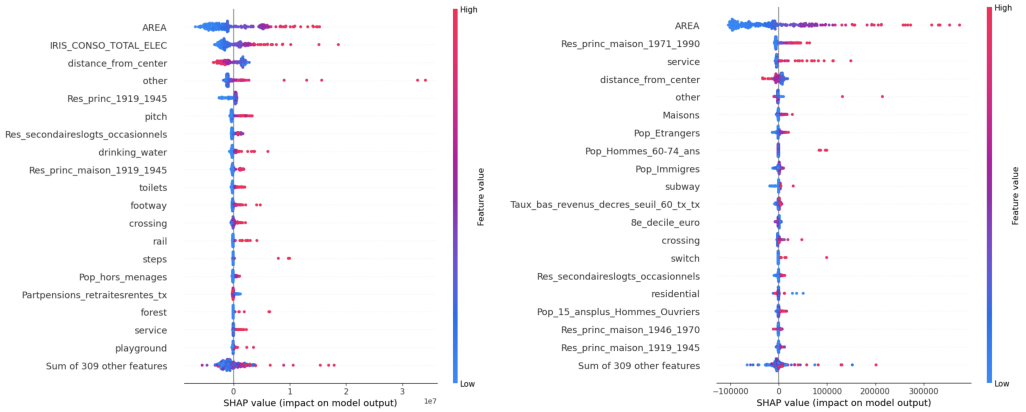

Figure 3.8: SHAP summary plot for the top 20 features - social (left) gaming (right)

The feature importance plots suggest that certain key features are crucial across both models and categories. The results for Social are more reliable than those for Gaming, given the greater performance error in the Gaming category. Here is a summary of the most important features:

- **AREA**: The total area of the IRIS cells is the most important feature of all; a smaller area reduces the traffic data, a larger area increases the traffic data.

- **Distance from the city centre**: The distance from the city centre is one of the few important features that has a clear negative effect on the predicted traffic data; a smaller distance from the city centre increases the traffic data, a larger distance from the city centre

reduces the predicted traffic data.

- **Electricity Consumption**: in particular the total consumption of electricity and of the tertiary sector. These are proxies for the presence of offices, buildings and, in general, urbanization in the area. For the social time series, a lower electricity consumption tends to reduce the predicted traffic data, a higher electricity consumption tends to increase the predicted traffic data. The evidence for gaming is less clear, but the models trained on gaming are less reliable.

- **Income**: Income in the IRIS cells is measured by the average income of the different deciles of the resident population; this can be a proxy of a central or rich area. For the social time series, a lower income in the area tends to reduce the traffic data predicted, a higher income tends to increase the traffic data predicted. The evidence for gaming are less clear on this feature, but again, the models trained on gaming are less reliable.

- **Presence of attractions (other feature)**: This is a residual feature, but in this category some important attractions of the cities have been comprehended. The contribution mechanism of this feature is similar to that of electricity consumption and income.

- **Urbanization: Number of buildings** (houses, number of primary residences, ...) **and of other relevant POIs** (subway, bus stops, crossing, etc...). A lower presence of buildings and other POIs related to the urbanization tends in general to reduce in general the traffic data, a higher presence of buildings and other POIs related to the urbanization tends in general to increase in general the traffic data.

Conclusions

The models trained on one city and evaluated on the same city showed mild differences in performance depending on the type of model implemented, almost no difference due to the time aggregation chosen for the target time series, but a considerable difference in performance when predicting time series from different categories of mobile applications (social, gaming). The Social time series seems to be an easier time series to predict based on the available features, probably due to the more predictable patterns of social network use. The synthetic time series generated for the Gaming category show a high prediction error, probably because these time series have much more variability (high and low peaks). The experiments show that both social and gaming categories still exhibit considerable prediction error.

Reduced models using the top 50 features based on different feature importance methods gave similar results to the complete models, suggesting that a smaller set of well-selected features are the most important ones for this prediction task (cell size, distance from the centre, average income of residents, level of urbanization of the area and electricity consumption) and must form the base set of features to be used for future implementations.

Since almost all of the demographic information used in this thesis

is related to the resident population in a certain area, a future development could be to incorporate additional metrics that reflect the dynamics of people movement within the city, such as car traffic data and tourist flows.

The map depicting the average errors made by models across various IRIS cells highlighted spatial correlation in these errors. The models used in this thesis did not consider factors like the time series of geographic neighbors, their prediction errors, or their characteristics to predict mobile traffic data consumption in specific areas. The only spatial feature included was the distance from the city centre, which may be insufficient to explain the spatial correlation among different areas in the city. Experiments using a limited set of significant features, such as area, electric consumption, and other services, were conducted to study their relationship with daily mobile traffic data aggregated into three periods (night, morning, evening). There was significant spatial autocorrelation, measured by the correlation in the residuals of a simple OLS model and the Moran's index. Spatial regression models demonstrated improved predictive performance in this context, suggesting that a cell's location and the characteristics of its neighboring cells can significantly impact its mobile traffic data consumption. Future work could comprehend models that account for spatial autocorrelation, for example by including additional features representing the characteristics of the nearest geographic neighbors (e.g., a weighted average of the characteristics of adjacent cells).

The models trained with data from one city showed limitations and a tendency to overestimate when used for predictions in other cities. This could be due to the structural differences existing in the average mobile

traffic data levels across cities, that cannot be explained by the set of features and models taken in consideration in this thesis, or, even worse, the mobile traffic data generating process might be completely different across different cities. One way to further improve the predictive accuracy of these models, beyond adding the aforementioned metrics that reflect the dynamics of movement within the city, may be to introduce proxies for consumer preferences that distinguish one city from another (e.g. coming from survey data). Another approach to address the limitations of these models performances is to adjust the predictions with a proportional constant factor. Experiments using as a proportional factor to adjust the predictions the ratio between the average data consumption in the training city and the predicted city yielded reasonable prediction errors (Comparable to the errors made by models trained and tested in the same city). However, the aim of these models is to generate time series without prior knowledge of the specific time series of the cities to be predicted, and there is no guarantee that the selected ratio will hold over time. The choice of the proportional factor to adjust the predictions of a model trained in one city to predict mobile traffic in other cities could be a focus for future research, and it would be better if this factor could be computed using readily available metrics.

Bibliography

[1] Molnar C.(2023). Interpretable Machine Learning, A Guide for Making Black Box Models Explainable

[2] Shapley L. (1953) A Value for n-Person Games. In: Kuhn, H. and Tucker, A., Eds., Contributions to the Theory of Games II, Princeton University Press, Princeton, 307-317.

[3] Martínez-Durive O. E. , Mishra S., Ziemlicki C., Rubrichi S. , Smoreda Z. , and Fiore M. (17 Jul 2023). The NetMob23 Dataset: A High-resolution Multi-region Service-level Mobile Data Traffic Cartography, arXiv - CS - Networking and Internet Architecture.

[4] Chen T., Guestrin C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. doi:10.1145/2939672.2939785

[5] Goodfellow I., Bengio Y., Courville A. (2016). Deep Learning. MIT Press.

[6] Kinsley H., Kukiela D. (2020). Neural Networks from Scratch in Python. Harrison Kinsley.

[7] Štrumbelj E. , Kononenko. I.(2014). “Explaining prediction models and individual predictions with feature contributions.” Knowledge and information systems 41.3 : 647-665.