MLNS
Traffic Prediction using GNN and spatio-temporal Neural Networks
Final Project Report

Elisabeth DAO
CentraleSupélec
elisabeth.dao@student-cs.fr

Zineb LAHRICHI
CentraleSupélec
zineb.lahrichi@student-cs.fr

Ilyass OUBAIK
CentraleSupélec
ilyass.oubaik@student.ecp.fr

Timothée RIO
CentraleSupélec
timothee.rio@student.ecp.fr

Abstract

Traffic forecasting becomes more important for the success of smart transportation, with many real-world applications such as intelligent route planning, dynamic traffic management, and smart location-based applications. Early intervention based on traffic forecasting is seen as the key to improve the efficiency of transportation system and to alleviate transportation-related problems. In recent years, machine and deep learning have helped build models to graph structures.Traffic prediction is a classical spatial-temporal prediction problem and due to the high non-linearity and complexity of data, deep learning approaches have attracted much interest in recent years. Graph neural networks have been introduced and have achieved state-of-the-art performance for traffic forecasting problems (traffic flow and speed forecasting, passenger flow forecasting problems...)

1. Introduction and Motivations

1.1. Emergence of intelligence transportation

Transportation systems are among the most important infrastructure in modern cities, supporting the daily commuting and traveling of millions of people. With rapid urbanization and population growth, transportation systems have become more complex.

Expanding cities face many transportation-related problems, including air pollution and traffic congestion.

In the development of smart cities and intelligent transportation, the increase of sensors to detect traffic states on roads, traffic surveillance videos and even smartphone GPS, more data have been collected. This gave a clear and interesting impulsion to project like ours, that process those data to obtain interesting outputs.

1.2. The interest of GNNs to resolve this spatial and temporal problem

The traffic state in a specific location has both spatial and temporal dependency. That is why traditional linear time series models cannot handle such spatio-temporal forecasting problems.

Graph neural networks (GNNs) have shown state-of-the-art performance in various applications for road traffic because it can capture spatial dependency, which is represented using graph structures. A road network is naturally a graph with road intersections as the nodes and road connections as the edges.

For this project, we have reviewed several papers that present GNN-related approached to traffic forecasting problems [5]

1.3. Our objectives

In this project, we aim at understanding the underlying dynamics behind road traffic in the city of Paris, through traffic forecasting. Our goal is to provide a GNN-based solution to predict traffic flow in various places, and altered versions of the same network.

To improve our prediction models, we also plan to use clustering methods to detect traffic patterns, for example between business and residential districts. This way, we hope to distinguish areas based on global and local dynamics.

Finally, our solution should enable to forecast traffic congestion at certain times and generalize to small alterations of the Parisian road network. Therefore, we should be able to measure the impact of small perturbations in the network such as edge removal (pedestrian transformation,

public works) and node removal.

2. Related Work

Machine Learning and Deep learning methods applied on graphs data structures have been widely used in traffic prediction. Machine learning methods are discussed in [2]. They include regression models, example-based models (k-nearest neighbors for instance) and kernel-based models (SVM, radial basis function).

Graph-based Deep Learning architectures are reviewed in [4], for a series of traffic applications : traffic congestion, travel demand, transportation safety, traffic surveillance, and autonomous driving. The advantages of both GNNs and other Deep Learning models (Recurrent Neural Networks, Temporal Convolutional Networks and Generative Adversarial Networks) are examined. In [3], the authors demonstrates that encode-decoder long short term-memory (LSTM) combined with graph-based methods (GNNs) is the state-of-the-art prediction technique.

For our project, we have mostly used [1], that proposes a Deep Learning architecture to solve the spatial-temporal traffic prediction problem. They also tackle sparsity and non-connectivity of graphs, encountered in real world applications. They propose a simple pre-processing step to obtain a connected graph and compute a proper weighted adjacency matrix as a base to implement models.

3. Data overview

3.1. Dataset

In this project, we leverage road traffic data captured from permanent sensors in the city of Paris. On the Parisian network, traffic measurement is mainly done through electromagnetic loops installed in the roadway. Open source data is available in the Opendata Paris website, which publicly provides updated data on daily traffic in the city of Paris (https://opendata.paris.fr)

The available data is raw and updated on a daily basis, over 13 sliding months. It is limited up to 3000 sections of lanes, described by their occupancy rate and flow rate. Although it does not allow to characterize the full complexity of the Parisian traffic, it will be enough to comprehend the dynamics and challenges of such a transportation system.

The provided features are :

1. **Occupancy rate** It corresponds to the time during which vehicles are present in a given section of lanes. It is expressed as a percentage of a fixed time interval (1h). The rate provides information on traffic congestion.

2. **Flow** It is the number of vehicles having passed the counting point during a fixed time interval.

3. **Timestamp** The observations are reported on an hourly basis. For example, the timestamp 2019-01-01 01:00:00 refers to the period from from January 1, 2019 at 00:00 to January 1, 2019 at 01:00.

4. **Flow characterization** Categorical feature describing the flow as fluid or saturated.

This dataset can be used via an API that allows records to be searched and downloaded by several criteria (counting date and time), flow characterisation, upstream and downstream node labels etc... For our project, we have first filtered the data to obtain information on one particular day in order for the sake of preliminary exploration.

However data collection on this API has some limitations. First we are not able to collect more that 10 000 points at a time, which is insufficient if we want to do specific queries on an entire day for instance (there is about 12 000 points per hour).

3.2. OpenStreetMap network

As a second step, we have explored another type of mapped data which relates to Paris City in order to work with more accurate and exhaustive information. Indeed, we wanted to supplement the data we already had to provide a complete overview of Paris mapping and to manipulate and implement more easily the theory of graph network.
For that, we have used Open Street Map Network , which is a Python package to retrieve, model, analyze, visualize street networks. We were able to query the map of Paris, and to easily plot a graph that represents each streets and intersections of the city.

Below is the plotted graph network we obtained from the city of Paris with this package :
To better understand the data, we decided to use the ”highway” feature that is a main used key to identify any kind of road or path. The value of the key helps indicate the importance of the highway within the road network as a whole. They are ordered from most important (motorway) to least important (service). We thought identifying those different types of road would help us to better understand the vehicles behaviours within the city and would give us hints for our initial clustering idea.
We obtain 13 types of roads :

- **Motorway** : Fast-traffic roads (with 2 or more running lanes plus emergency hard shoulder) (clear blue)

Figure 1: Graph from Paris obtained with OSMNX Python Package

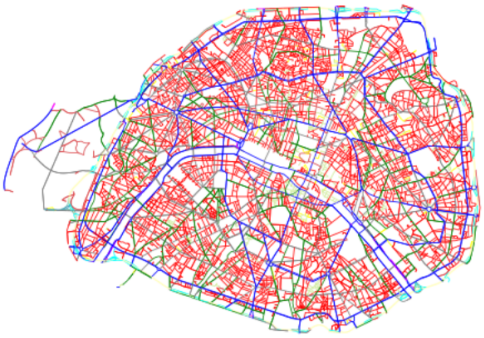

Figure 2: Highway types in Paris (legend above)

- **Motorway link** : Leading from/to a motorway road
- **Trunk** : Main roads connecting major towns or cities (yellow)
- **Trunk link** : Leading from/to a trunk road
- **Primary** : Most important roads in a city (dark blue)
- **Primary link** : Leading from/to a primary road
- **Secondary** : Next most important roads in a city (gray)
- **Secondary link** : Leading from/to a secondary road
- **Tertiary** : Again next most important roads in a city (green)
- **Tertiary link** : Leading from/to a tertiary road
- **Unclassified** : Least important roads in a city
- **Residential** : Roads which serve as an access to housing (red)
- **Living street** : Residential streets where pedestrians have legal priority over cars (speeds are kept very low)

We have coloured the different highway types in Paris with the above legend:

3.3. Pre-processing

The pre-processing step aims to solve three problems we have met when working with our original dataset with traffic data :

1. Traffic data is not available for all roads
2. The graph we create with our original data does not form a connected dense graph.
3. The graphs obtained with OSMNX and the OpenData don't have perfectly matching node locations, even though they point to the same streets.

   To solve these issues, we need to implement both a matching and completion algorithms.

3.3.1 Matching algorithm

To solve the matching problem, we implemented an algorithm to combine the two maps we obtained.

For that, we created a graph from our OpenData json file with the help of the longitude and latitude for the nodes. Every edge was then provided with different available attributes Below is the graph we obtained :

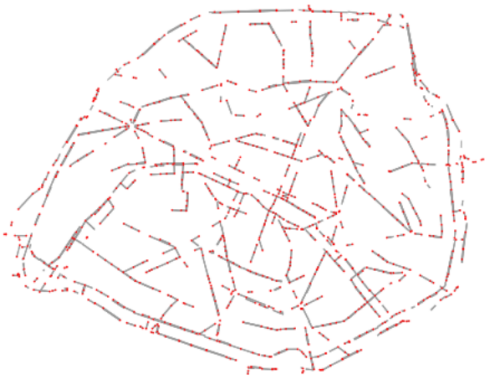

Figure 3: Graph obtained with OpenData

We note that the graph is not complete, nor dense or connected. That is why we decided to combine it with the OSMNX map. However, all nodes did not correspond accurately in our two cards. Thus we used a very simple algorithm where we make every node of our first graph correspond to the nearest node by distance of our second graph.

3.3.2 Connected Subgraph Search Algorithm

We decided to work on a smaller subgraph to optimize our calculations and do our experiments on different scales of the map. In order to obtain a connected subgraph, we developed an algorithm that extracts a set of nodes from a given graph, based on the BFS search algorithm. Basically, Bread-First sampling is a node sampling algorithm which runs through the graph from a central point and build little by little a connected subgraph. It is a widely used approach for sampling large complex networks that consist of a tremendous amount of nodes and edges, which make them difficult to extract and analyze. Its main advantage over random walks and other exploration techniques is that a BFS sample is a plausible graph on its own and therefore, we can study its topological characteristics.

We thus obtained a connected subgraph of Paris Center around Rivoli Street, as shown in 4

Figure 4: Subgraph of Paris roads centered on Rivoli Street

Next step was to integrate the traffic data that were available on certain roads from our original dataset to our new pre-processed graph obtained with the combination of our two maps.

On this illustration, you can see in blue the original Paris map, collected with the OSMNX python package. You will also find in green the road portions on which we managed to get traffic data from the Paris open data source. In red are the traffic data that we had from the open data source but that do not overlap with our OSMNX map. We decided to still consider those red portions because they seem coherent with the map we had as they are located as a continuum of the blue osmnx roads.

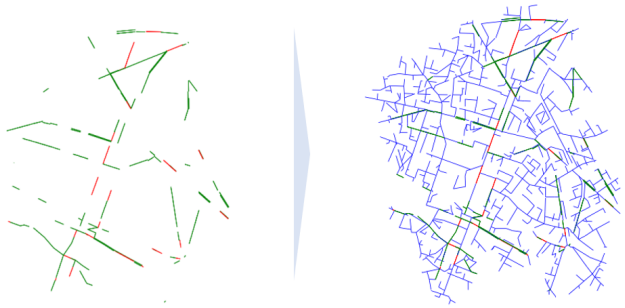

Figure 5: Maps correspondance

3.3.3 Graph completion via interpolation method

In order to treat the problem of missing data on our current map, we used a linear interpolation method. Basically with this method, all traffic data are assigned to their nearest roads and weighted with a Gaussian kernel as described below. Indeed, we reasonably consider that traffic flow is nearly the same for the close neighbourhood of our data-filled roads. Our pre-processed work at this stage is inspired from [1].

Basically, we form the completed weighted adjacency matrix as follows :

$$w_{ij} = \begin{cases} w_{i'j'} * exp(dist/\sigma) & \text{if } i,j \text{ are neighbours} \\ 0 & \text{if } i,j \text{ aren't neighbours} \end{cases}$$

We applied this formula for both the flow and ratio of the roads with $dist$ being the distance between an edge and its nearest edge for which we have information, and σ fixed at 0.005.

4. Traffic Pattern Labelization

4.1. Supervised Method : Naive Bayes

4.1.1 Methodology

Based on the already labelized data we had thanks to the OSMNX library on the different types of Paris roads (see Figure 2), we decided to build a model that can predict the types of roads thanks to the different characteristics we had on the roads.

These characteristics are the following features :

- **Maxspeed** : the maximum speed at which a car can travel
- **Oneway** : if it's a one-way road or not (False or True)

- **Topology** : topological feature mainly based on the length of the road

4.1.2 Algorithm and results

In order to build this first supervised prediction model, we have used a simple Naive Bayes classifier, which is a probabilistic machine learning model used for classification task based on the Bayes theorem. With this method, we are estimating the conditional probability of the class label, given the observation.

Thanks to this classifier, we manage to predict the types of roads based on its traffic and topologic features.

Below is an illustration of what we obtained thanks to our model applied to the Champs Elysées zone :

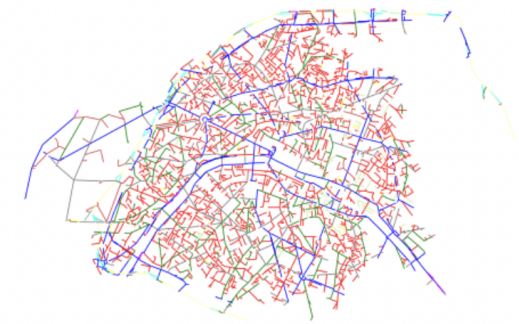

Figure 6: Type of roads prediction around Champs Elysées

4.2. Unsupervised Method : Clustering

4.2.1 Methodology

The aim of this step was to try to distinguish areas that behave similarly in order to detect traffic patterns and improve our prediction results. For instance, we hoped to distinguish dynamic neighbourhood that would be for instance business districts and quiet neighbourhoods that would be more residential.

4.2.2 Features description

We considered several features that are the traffic characteristics on the roads we used to distinguish our clusters :

- Flow : the number of vehicles in a lane at fixed time intervals
- Ratio : the time during which vehicles stay in a lane at fixed time intervals

Using these features on a specific interval at first would allow us to distinguish three clusters that behave differently, from the most congested roads to the less.

4.2.3 Algorithm and Results

We used the Kmeans module of the Sklearn libray to try to distinguish the clusters among the different roads regarding the features described above.

We decided to display three clusters that you can see on the figure below in three colours (blue, red and green). We have represented in bigger size the major boulevards

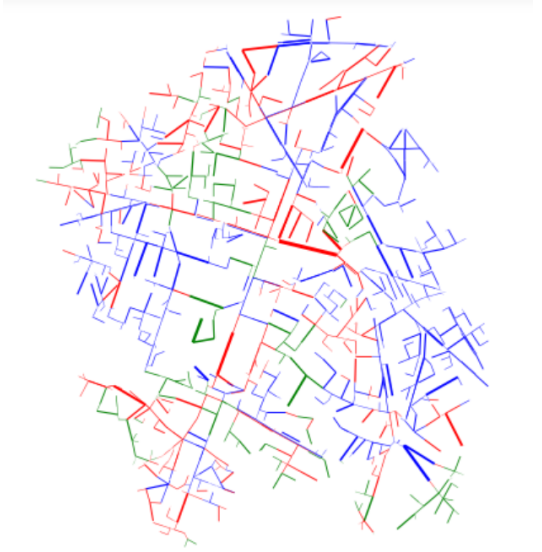

Figure 7: Flow distribution in training set

In this graph, we can distinguish three types of areas :

- In red, are the roads with the less flow and ratio, which might represent residential areas or event pedestrianised roads. We can recognized the Rivoli street that horizontally crosses the graph from left to right in red, since its is now entirely reserved for pedestrians and bicycles and definitely closed to cars.
- In green are the most congested roads, with a higher flow and ratio. It represents the most frequented areas of Paris. We can easily identify the Place de la Concorde which is often congested as it is a major crossing point in Paris.
- In blue are the roads with average flow and ratio

Of course this result is limited by the little number of data we had to solve this problem. However, it gives consistent results with the reality.

5. Traffic Prediction

5.1. Methodology

The methodology we used to predict the traffic (ie : flow and ratio) was to use a KNN algorithm (chapter before) in

order to select spatiotemporal correlation stations with the test station at first, and then use a two-layer LSTM network to predict traffic flow, respectively, in selected stations.

5.2. Feature description

We have chosen to have a 3 clusters dataframe in order to apply the methodology.

Since we have a limited database (around 10k points) for only 2 days. we have chosen to train our models on date 2022-02-27 at 9:00 and 10:00 p.m and then test it at the same time the next day for each cluster

5.3. Pre-Processing

LSTMs are sensitive to the scale of the input data, specifically when the sigmoid (default) or tanh activation functions are used. It can be a good practice to rescale the data to the range of 0-to-1, also called normalizing. We can easily normalize the dataset using the MinMaxScaler preprocessing class from the scikit-learn library.

For each cluster here is the the distribution of the features:

5.3.1 Distribution of flow per cluster

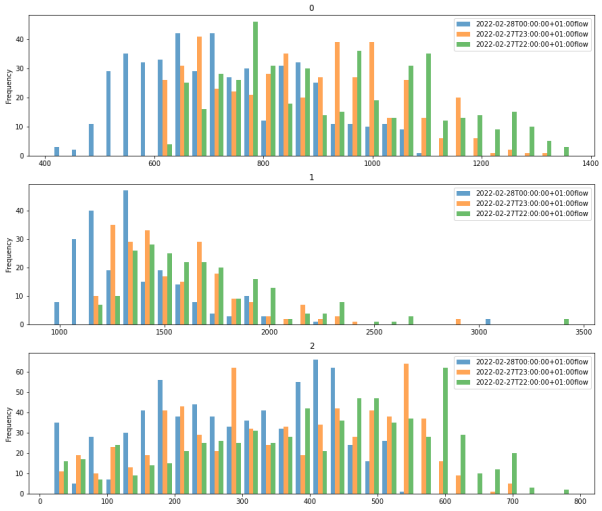

Figure 8: Flow distribution in training set

5.3.2 Distribution of ratio per cluster

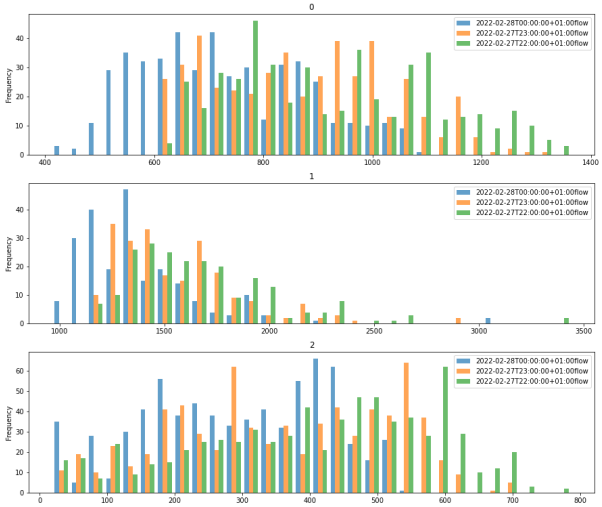

Figure 9: Ratio distribution in training set

5.4. Model description

RNN is a neural network that is specialized for processing time sequences.

Different from conventional networks, RNN allows a "memory" of previous inputs to persist in the network internal state, which can then be used to influence the network output. Traditional RNN exhibits a superior capability of modeling nonlinear time sequence problems, such as speech recognition, language modeling, and image captioning.

However, traditional RNN is not able to train the time sequence with long time lags. To overcome the disadvantages of traditional RNN, LSTM has been proposed. LSTM is a special kind of RNN, designed to learn long term dependencies.

The LSTM architecture consists of a set of memory blocks. Each block contains one or more self-connected memory cells and three gates, namely, input gate, forget gate, and output gate. Input gate takes a new input from outside and process newly coming data. Forget gate decides when to forget the previous state and thus selects the optimal time lag for the input sequence. Output gate takes all results calculated and generates output for LSTM cell.

5.5. Pipeline description

in this section we will describe the pipeline we used for out traffic prediction.

1. Select mostly related stations (KNN)
2. pre process flow/occupationrate with MinMaxScaler
3. Train traffic flow with LSTM network, respectively, in selected clusters on date 2022-02-27 (see diagram)
4. Predict traffic flow with LSTM network, respectively, in selected clusters on date 2022-02-27 (see diagram)
5. check performence metric per cluster and scatter plot (flow,occupationrate) vs predicted (flow,occupationrate)

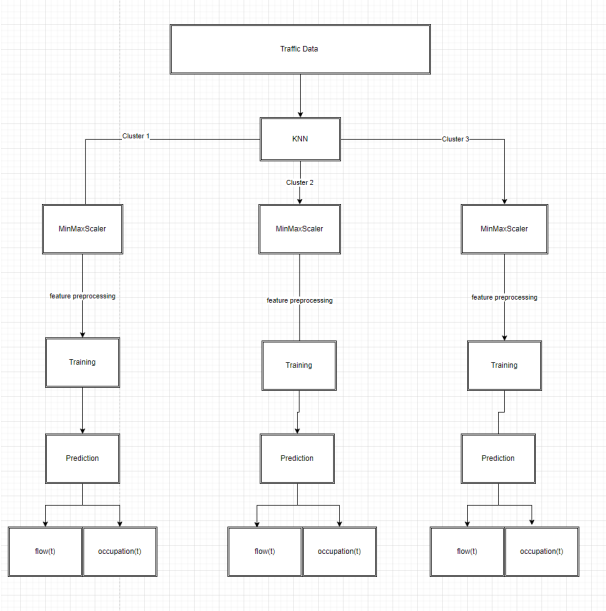

Figure 10: Pipeline diagram

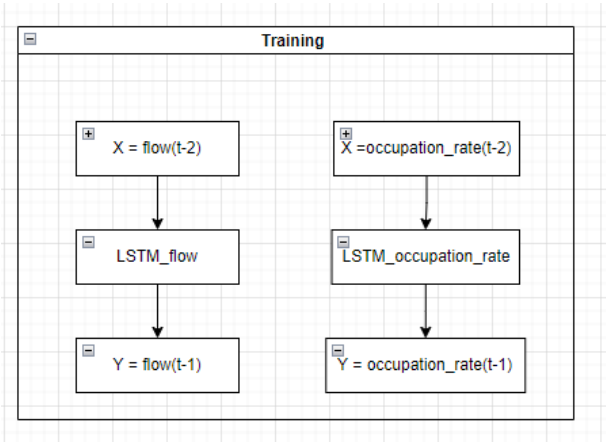

Figure 11: Pipeline diagram

5.5.1 Distribution of ratio per cluster

5.6. Performance metrics

In order to evaluate the prediction we have made we have chosen 2 performance metrics to do so :

5.6.1 Root Mean Square Error (RMSE)

Mean squared error (MSE) measures the amount of error in statistical models. It assesses the average squared difference between the observed and predicted values. and we have used it in order to measure performance of our model

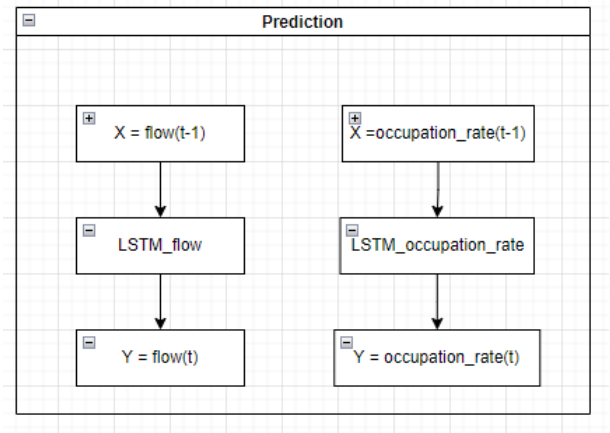

Figure 12: Pipeline diagram

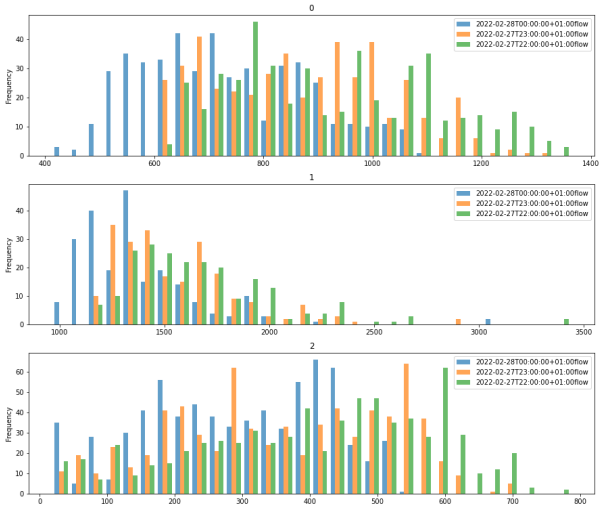

Figure 13: Ratio distribution in training set

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$

5.6.2 R-squared (R2)

R-Squared (R^2 or the coefficient of determination) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable. In other words, r-squared shows how well the data fit the regression model (the goodness of fit)

$$R^2 = 1 - \frac{\left(y_i - \hat{y}_i\right)^2}{\left(y_i - \bar{y}_i\right)^2}$$

5.7. Traffic prediction result

In this section we are exposing the couples (flow,ratio) predicted versus reality in test part (i.e trying to predict the flow, ratio in at the 28th of February date) for each cluster , keep in mind that it's scalled from 0 to 1 using MinMaxScaller from sklean library

In red are the predictions and in blue are the real values

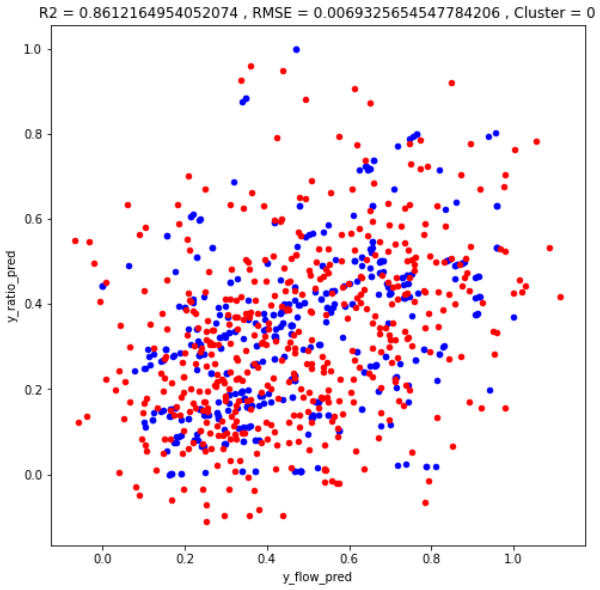

Figure 14: Prediction comparison for cluster 0

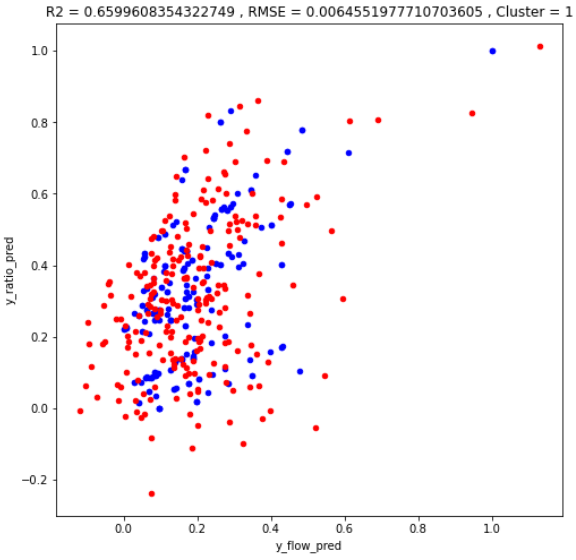

Figure 15: Prediction comparison for cluster 1

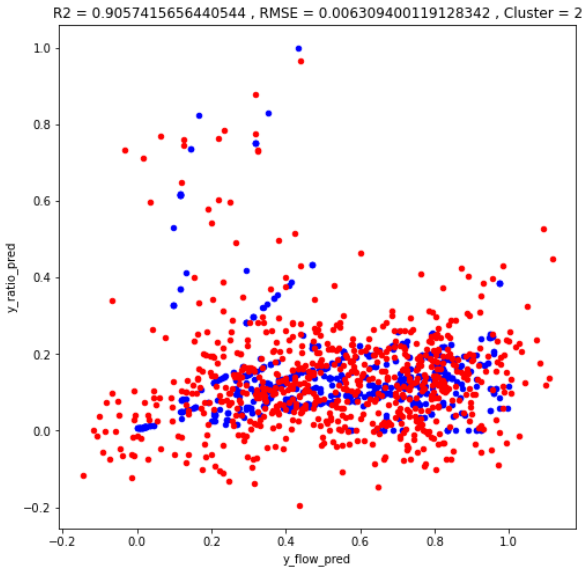

Figure 16: Prediction comparison for cluster 2

5.8. Conclusion

In this project, we proposed a spatiotemporal traffic flow prediction method combined with KNN and LSTM. KNN is used to select mostly related neighboring stations (in our case we took 3 clusters) that indicated the spatiotemporal correlation with the test station. A LSTM network was applied to predict traffic flow and occupation rate , respectively, in selected stations (ie : clusters). LSTM is able to exploit the long-term dependency in the traffic flow data and discover the latent feature representations hidden in the traffic flow, which yields better prediction performance. The final prediction results in test station are obtained by affecting the station to right cluster then predicting using LSTM trained on that cluster. We evaluated the performance of our model with real traffic data provided by Paris traffic website . The results show that proposed model(KNN + 2 layer LSTM) is a good structure for traffic prediction .but Since the traffic flow data is affected by weather, incident, and other factors, the impact of these factors we can improve it's structure . this part can be kept for further research.

References

[1] Rongzhou Huang, Chuyin Huang, Yubao Yubao Liu, Genan Dai, and Weiyang Kong. Lsgcn: Long short-term traffic prediction with graph convolutional networks. pages 2327–2333, 07 2020. 2, 4

[2] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey, 2022. 2

[3] David Alexander Tedjopurnomo, Zhifeng Bao, Baihua Zheng, Farhana Choudhury, and A. K. Qin. A survey on modern

deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020. 2

[4] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, and Hui Xiong. Coupled layer-wise graph convolution for transportation demand prediction, 2020. 2

[5] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. *CoRR*, abs/1709.04875, 2017. 1