Master in Data Science for Decision Making

BiciMAD Bike-Sharing System Dataset Creation and Preliminary Hourly Demand Prediction using Machine and Deep Learning Approaches

Authors:

Roset Cardona, Javier

Gutierrez Noelle, Erika

Contents

1 Introduction

By 2050 it is estimated that 68% of the world's population will live in urban areas [15]. A key problem city governments have to deal with is traffic. It has been shown that bikeshare systems can reduce traffic congestion by as much as 4% [7]. Furthermore, these systems have been proven to reduce the greenhouse gas emission in a city [17, 9]. Meaning, Bikeshare systems are a solution to ease the congestion in a city in a sustainable way. The rate of the number of cities with bikeshare systems has been growing rapidly since 2007 [4]. According to The Medding Bikeshare Map, since 2022 there are 1,590 cities with bikeshare systems. Bikeshare systems with bike docking stations still outnumber bikeshare systems with dockless bikes. A key issue operators of bikeshare systems with docking stations must keep in mind is station rebalancing. Station rebalancing is restoring the number of bikes in each station to an amount that minimizes the instances of impossibilities, in a manner that minimizes the cost associated with deploying station rebalancing fleet vehicles. Impossibilities are instances when a bikeshare customer wants to rent a bike but no bikes are present at the station or when a bikeshare customer wants to return a bike to a station but no docks are available. The first step in optimizing rebalancing operations is to predict the demand for bikes for each station. Forecasting this demand is the topic of this project. In this paper we describe how we combined data from several sources to create a unique feature set that we use to predict hourly demand for every station in Madrid's bikeshare system using state-of-the-art machine and deep learning approaches. Our work shows that these approaches perform well in understanding the bike demand flows within the city and provide a solid basis for further research.

2 Objective

The objective of this project consists on the creation of a dataset of BiciMAD bike-sharing system data oriented to bike demand prediction that covers from 2019 until the end of 2022, as well as a preliminary prediction approach using state-of-the-art techniques such as XGBOOST and GCNN.

2.1 Scope

For the Barcelona School of Economics (BSE) Master in Data Science for Decision Making's Masters Thesis Challenge we collaborated with Innova-TSN, a leading consulting company in the application of analytics and the design of customised solutions with offices in Madrid, Barcelona, Santander, London, México City and Bogota. This company participated in the 2022 SAS Global Hackathon where they developed a solution to optimize BiciMAD by creating forecasting models that provided the number of bicycles demanded and returned at each BiciMAD station for each day and hour and estimating the number of bicycles to replace or remove from each station using optimization algorithms. They proposed to us to improve their demand prediction results, and as guidance suggested the following:

- Adding new variables related to demand such as hour weather variables per station, imputation of missing values, and aggregation of stations.
- Optimizing the training of the algorithms used (ARIMA, XGBOOST, LightGBM, LSTM)

- Test other forecasting algorithms

Given the project's time constraints and the intricate nature of the problem at hand, it was not feasible to accomplish all aspects within the designated timeframe. First, we faced a lot of problems when dealing with more recent BiciMAD data (Innova-TSN only used data until 2021) which had different formats, as well as other problems when trying to add variables such as hour weather variables. For this reason, the concrete tasks we have done are:

- Creation of a coherent BiciMAD dataset, with extra features such as weather measurements per hour assigned to each station, and city-wide holidays. The procedures developed to merge all data sources allow the creation of the data frame from the start of 2019 into the future, and are robust to differences in data files, as well as inconsistencies in the data sources at the moment this report has been written.
- Preliminary prediction study over the dataset gathered using state-of-the-art techniques (XGBOOST and GCNN).
  - Prediction of plugs and unplugs in each station at each hour. Plugs are the bikes that have arrived at the station and unplugs are the bikes that have left the station.
  - In terms of the prediction using GCNN, we will only use the static graph approach.
- Hyperparameter optimization of an XGBOOST model using a different framework and larger parameter search space than the one used by Innova-TSN.
- Analysis of the results obtained.

The tasks that haven't been done:

- A careful data analysis to determine the quality of the final dataset.
- Complete feature engineering to exploit all the potential of the data in terms of predicting power.
- Research to find the best model to predict demand in BiciMAD bike-sharing system.

2.2 Literature Review

The literature of bikeshare systems mainly begins in 2010 with the focus of the field turning to more towards demand prediction, rebalancing, and redistributions of bikes among stations in 2016 [16]. However, one of the first and most cited studies for predicting station level bike demand was published in 2010. In this paper, Kaltenbrunner et al. analyze the spatial and temporal distributions of cyclic mobility data mined from the website of Bicing, Barcelona's bikesharing system [8]. They also predict the hourly demand at the bike station level through the use of an ARMA statistical model. As both research in bikeshare bike demand and machine learning grew, so did the research of applying machine learning methods to predict bike demand, with Random Forests being one of the most popular algorithms [1]. Post 2016, the subject matured to the use of deep learning techniques as the standard. The types of methods that have been used include different configurations of

Long Short Term Memory, Deep Neural Networks, Recurrent Neural Networks, Gated Recurrent Neural Networks, and Graph Convolutional Neural Networks [1]. Spatial Temporal Graph Convolutional Neural Network (STGCNN) based deep learning model architectures provide one of the best performances and has been the subject of the majority of the recent literature in this field. In these studies, the bikeshare system is represented using a fully connected undirected graph object where the stations are the nodes. The adjacency matrix associated with this graph object can contain various measures of correlation between the stations. Most often spatial distance is used but it can also be average trip duration or average demand between stations [11, 14, 5]. In the first layer of the network architecture a convolution using this adjacency matrix is applied to extract the signals of correlation among the stations. This information then goes through subsequent network layers to finally output the demand for each station at the future time periods. There are various types of layers that the GCN output can be passed to to achieve this. Lin et al. passed the result of the GCN to a LSTM block then a FFN block. Guo et al. use a GCN to GRU encoder decoder structure. Kim et al. provide the GCN output to a fully connected layer. The most sophisticated approach we encountered was published by Liang et al. in November 2022. In this paper, the authors propose a domain-adversarial multi-relational graph neural network (DA-MRGCNN) that combines bikeshare use data and historical demand data from public transport and ride-hailing services in New York City to predict hourly demand for bikeshare bikes [10]. Based on this research we realized the spatial correlation of the bike stations in the system was a key variable to take into account and a GCNN was a model architecture well equipped to handle the non-Euclidean nature of the bikeshare system spatial structure. Therefore, we decided the model was worth testing for our task.

3 Methodology

The main steps of this project have been the decision and implementation of the model, the creation, improvement, and processing of the BiciMAD dataset, and the training of the models implemented.

3.1 Models

Predicting station level demand for bikes is not trivial as it is dependent on a variety of highly dynamic factors such as weather and city-wide events. Furthermore, demand for each station cannot be studied in isolation as it is dependent on hidden spatial and temporal correlations between other stations in the system. For our project we will work with 3 models, a baseline, an XGBOOST model, and a GCNN.

3.1.1 Baseline

For plugs and unplugs at the current time period, the baseline will predict the value each station had at the same hour in the previous week, taking advantage of the weekly seasonality of the data. The baseline will serve us to compare the performance of the more complex models (XGBOOST and GCNN) with a trivial solution and know if the other proposed models are capturing other trends.

3.1.2 XGBOOST

We chose to use an XGBOOST because it provided the best results for Innova-TSN in their hackathon solution. This model can be used on a wide variety of structured data applications with premier prediction performance and computational speed. It can be used for time series prediction as long as the training and test sets are made keeping chronology into account. XGBOOST stands for extreme gradient boosting. This is an ensemble model that computes multiple models (usually decision trees) in a sequential fashion, using the the errors from the previously estimated model to inform the creation of the new one. Moreover, during the training process, XGBOOST calculates the gradient of the loss function with respect to the model's predictions. The gradient indicates the direction to update the model to minimize the loss and these values are used to guide the building of new models, hence the "gradient boosting" in the name.

3.1.3 Graph Convolutional Neural Networks (GCNN)

As previously commented, the demand for a bike-sharing system is very dependent on hidden spatial and temporal correlations between other stations in the system. If we think about it, we can imagine a bike-sharing system as a network where each station is a node, and they are connected between them with stronger or weaker edges. For this approach, the best-suited models are GCNNs which are well-suited for capturing such spatial relationships by leveraging the graph structure of the bike stations network and making use of the geospatial relationships between stations. In terms of temporal dependencies, GCNNs can also capture them thanks to their ability to model information propagation across graph nodes over time.

Another very interesting feature of GCNNs is that they can handle dynamic graphs, which means they can be robust to the incorporation of new nodes over time, or the removal of them. And even better, they can be used to assess the effect of adding or removing some nodes. This can help optimize resource allocation, improve operational efficiency, and enhance the user experience by enabling better planning and management of bike availability and redistribution strategies. For all these reasons, GCNNs have the potential to provide more accurate and robust predictions of bike demand in a bike-sharing system than other approaches.

For this project, we will only use static graphs as we are only doing a preliminary prediction study. This will force us to create a subset of the original data where we will need to remove all stations that are not constant over the time span we are studying. Also, to introduce the geospatial relationships among stations, the GCNN will need an adjacency matrix to understand how the nodes are connected and which weights the edges have. In terms of the tools used to build a GCNN, PyTorch has developed an extension of their package to work with geometric structures including graphs called *pytorch_geometric_temporal*[13]. More precisely, the GCNN that we used is an implementation of the Attention Temporal Graph Convolutional Cell. More information can be found in the package[12] and in the paper [2].

3.2 Data

In this section, we will do an overall comment on all the processes regarding the data, from the data sources and their characteristics to the final dataset and its features.

3.2.1 Data sources

We made use of two open databases and the Google Maps API for the data used in our project. The first database was hosted by the Empresa Municipal de Transportes (EMT) of Madrid. EMT Madrid is a public limited company, owned by the Madrid City Council, that is in charge of managing and operating Madrid's public transportation systems, including BiciMAD. In this database we obtained the status of each bike docking station for every day and hour as well as the recorded bike trip transactions from January 2019 to December 2022. Other than identifier variables, the most important variables from the data for the status of each bike docking station were:

- light (the level of occupation)
- activate (whether the station was active)
- no_available (whether the station was disabled or enabled)
- total_bases (the total number of bike docks at the station)
- dock_bikes (the number of bikes currently docked at the station)
- free_bases (the number of available bike docks at the station)
- reservations_count (the number of bikes currently on reserve at the station)

In the trip transaction data we obtained the following information:

- names of the stations where the bike was unlocked and unlocked
- timestamp of when the bike was unlocked and locked
- the geolocation of where the bike was unlocked and locked
- the address of the station where the bike was unlocked and locked
- The duration of the trip from the unlock station to the lock station in minutes

The second database was hosted by DatosAbiertos, an initiative related to open data in the city of Madrid, Spain. This initiative aims to promote transparency, innovation, and citizen engagement by making public data freely available and accessible to the general public. DatosAbiertos Madrid focuses on collecting, organizing, and publishing datasets from various government departments and agencies in a standardized and machine-readable format. In particular, we were interested in the weather data, that was gathered from the Sistema Integral de la Calidad del Aire del Ayuntamiento de Madrid. This system has several databases of hourly and hourly data starting from 2019, this specific source is the one that prevented us from using data previous to 2019, as we wanted to use hourly weather data per bike station. The weather magnitudes contained in this database are:

- Wind speed (code in database: 81)
- Wind direction (code in database: 82)
- Temperature (code in database: 83)
- Relative humidity (code in database: 86)
- Barometric pressure (code in database: 87)
- Solar radiation (code in database: 88)
- Precipitations (code in database: 89)

As Madrid is a big city, we did not want to assume that all stations had the same conditions at one moment in time. It is a possibility that in one part of the city it is raining, and on the other side it is not, which could influence bike demand. For this reason, we used the coordinates of bike stations and weather stations to allocate at each moment in time the measurements of the closest weather station to each bike station as can be seen in Figure 3 for May 2021. Here we faced one problem with the recording of data, as functional stations changed every month and at some months the stations did not gather information during all days for some magnitudes so we had to filter each month by the active stations per magnitude to then assign the measurements to the closest bike station.

We also used DatosAbiertos to get the working days information, as it contains a working calendar with all the holidays indicated since 2013.

Finally, we used Google Maps API to get the data for the adjacency matrix of the GCNN. The adjacency matrix contains the walking distance in meters from each bike station to every other bike station in the bike-share system. We used the API because it allowed us to get walking distance, which is a more realistic distance than the geospatial distance, as this one goes through buildings and is not the normal route of a user. Then, we rebuilt the matrix as the inverted distance adjacency matrix, as we wanted to express stronger relationship between closer nodes than further nodes.

3.2.2 Feature Engineering

The feature engineering applied to create the final database consisted only on creating cyclical variables using *feature_engine* package to all of those variables that had a cyclic behavior. The variables

- month
- day
- hour
- weekday
- week_of_year

were transformed into a combination of sine and cosine expressions which are able to capture the cyclic behavior.

A part from this, we have also lagged some variables. For example, for the GCNN, as the input consists on the data over all stations per one week to predict the plugs or unplugs over the next week, we have considered lagging the weather variables so that we are using the forecast (assuming it is very good) to predict the number of plugs and unplugs.

3.2.3 Final Dataset

We combined the data outlined above into a final dataset containing a feature set and the number of plugs and unplugs for every BiciMAD bikeshare system at every day and hour from January 2019 to December 2022.

Plugs being instances when a customer locks a bike and unplugs being instances when a customer unlocks a bike from the station. At the end of 2022 there were 266 stations in the BiciMAD bikeshare system. The final dataset contained 8,185,250 observations of 33 features. The final set of features that was used to train our models were:

- activation status of the station (activate)
- number of active reservations (reservations_count)
- total number of bike docks the station contains (total_bases)
- the number of available docks currently at the station (free_bases)
- whether the station was available or not (no_available)
- the number of bikes currently docked at the station (dock_bikes)
- latitude
- longitude
- Temperature (83)
- Relative humidity (86)
- Barometric pressure (87)
- Solar Radiation (88)
- Precipitations (89)
- the sine of the month number (month_sin)
- the cosine of the month number (month_cos)
- the sine of the day number (day_sin)
- the cosine of the day number (day_cos)
- the sine of the day number (hour_sin)
- the cosine of the hour number (hour_cos)
- the sine of the weekday (weekday_sin)
- the cosine of the weekday (weekday_cos)
- the sine of the week number of the year (week_of_year_sin)
- the cosine of the week number of the year (week_of_year_cos)
- the cosine of the wind speed value (wind_cos)
- the sine of the wind speed value (wind_sin)
- 1 if the level of occupation at the station is low otherwise 0 (light0)
- 1 if the level of occupation at the station is high otherwise 0 (light1)
- 1 if the level of occupation at the station is medium otherwise 0 (light2)
- 1 if the level of occupation at the station is unavailable otherwise 0 (light3)
- 0 if the day is on the weekend or a holiday 1 otherwise (work_day_indicator)
- 1 if days fall within a timeperiod associated with COVID-19 (15/3/2020 - 26/4/2020) 0 otherwise (covid_indicator)
- The number of unplugs for the same hour and weekday of the previous week (unplugs_week_lag)
- the number of plugs for the same hour and weekday of the previous week (plugs_week_lag)

When we trained our models, we scaled all these features by removing the mean and scaling to unit variance.

When deciding how to code the GCNN we realized that the model we had as a template could only handle a static graph object. Static meaning that the number of nodes and edges does not change over time. Unfortunately,

the number of bike stations in the BiciMAD system is not constant over time. Therefore, we only trained the GCNN using the stations that were present during the entire time period of the data we collected. This meant we trained the GCNN with 170 stations out of the 266 total stations in the system.

3.2.4 Data Problems

As we were processing the data we encountered many errors. Due to time constraints caused by these errors we were unable to do a proper exploratory data analysis post processing before using the data for training. Specific information about the errors encountered can be found in the Data Errors section of the appendix and on our Github [6].

3.3 Training Process

After putting together the data, it is time for training the models and analyze how they perform over the dataset. In this section, we will get into the specifications of how we prepared the data for the models, and what splits were created.

3.3.1 Training sets creation

We tested our model on two time periods, one time period was from January 2nd, 2022 to December 31st, 2022. The second time period was from June 20th, 2021 to July 3rd 2021. For the rest of the report we will refer to these as the 2022 test set and the Innova-TSN test set respectively. This second time period was upon request by Innova-TSN to see if our model had similar accuracy in predicting a similar time period that they tested their models on. The time period Innova-TSN tested their models was from June 24th, 2021 to June 30th, 2021. We did not use the exact same time period as Innova-TSN because during the model development process, we realized our GCNN model needed to have complete weeks as input and output, as it was able to differentiate better among the days of the week if the input order was consistent (Monday was always in the same position). For this reason, and to be consistent, we decided to design the splits so that they matched with weeks that start on Sunday and end on the next Saturday.

For the XGBOOST model we took two approaches to conduct training. One approach was to use all of the data prior to each testing dataset start date and use it to train the model in one go. The other approach was using a rolling window approach for the training set, whereby instead of giving the model the training data all at once, it would be given, every week prior to the testing dataset start date in a sequential fashion. We considered this approach because we wanted the model to learn the sequential nature of data tied to time. It was suggested to us that a rolling window approach may perform well for this since the relationships in our data between the target and the predictors may change over time and therefore a rolling approach would be able to better capture these changes and give us better predictions. We implemented this approach but it did not give us better predictions than our first approach. Futhermore, we accounted for the weekly seasonality by using an autoregressive approach through the inclusion of a week lag variable for the number of plugs and unplugs.

3.3.2 GCNN training set specificities

Graph convolutional neural netoworks use graph structures to understand geospatial relations between nodes. For this reason, the input for these models needs a specific format. The inputs and outputs of the model are the following:

- Adjacency matrix. This will get separated into the following elements (for pytorch geometric temporal package):
  - Edges index: size: (number of nodes*number of nodes, 2). Expresses the connections between edges
  - Edges weights: size: (number of nodes*number of nodes, 1). Expresses the strength of the relationship between those 2 nodes.
- size: (number of nodes, number of nodes)
- input: size: (number of time steps in, number of nodes, number of features)
- output: size: (number of time steps out, number of nodes)

Where in each time step you have all information of the network at that time. So when preparing the training sets for this model we needed to take all of this into account and shape the training and test sets accordingly. Also, as the whole infrastructure implements an autoregressive approach, it is very important to order the data properly to make sure that all information is consistent so that the model can learn the inherent relationships in the dataset.

3.3.3 Hyperparameter search

For obtaining the best hyper parameters of their XGBOOST model, Innova-TSN used a framework called HyperOpt. Hyperopt is a distributed asynchronous hyperparameter optimization framework written in Python. It makes use of either a Random Search or Tree of Parzen Estimators algorithm to explore the user-defined parameter search space minimizing or maximizing a user defined objective function for a user-defined number of trials. In each of these trials a value from the parameter space is chosen stochastically and used to train and test the model. If the search algorithm is Tree of Parzen Estimators then a probabilistic model based on the evaluations of these hyperparameter configurations is built that informs the selection of the next set of hyperparamters to test. This probablistic model allows the hyperparameters to be chosen in a exploratory and exploitative manner throughout the number of trials. The probabilistic model exploits the regions of the search space that have shown promise in previous trials, while still exploring other areas to discover potentially better configurations. The Random Search algorithm explores the parameter space randomly and does not use past trials to inform the selection of hyperparameters in subsequent trials. For our model hyperparameter optimization we used a python package called Optuna. Similar to HyperOpt, Optuna explores the user-defined hyperparameter search space for multiple trials, in what Optuna refers to as a study. A study is the optimization based on an objective function and a trial is a single execution of the objective function. Optuna has more parameter search space algorithms than HyperOpt and has the option to use pruning algorithms. Pruning algorithms automatically stop unpromising trials at the early stages of the training.

For their implementation, Innova-TSN explored the parameters mentioned in Table 1 using the minimization of the squared error as their objective function. It is unclear what search algorithm was used since based on code screenshots shared with us, they allowed HyperOpt to automatically pick it for them. In our hyperparameter optimization we also used the minimization of the squared error as our objective function. We used the Tree of Parzen Estimators algorithm to search the parameter space and the Hyperband algorithm for pruning. Each of our studies consisted of 100 trials. We explored more parameters than Innova-TSN; the parameters we searched for and the results for each model and test set can be seen in Table 2 and Table 3. Most of the parameters we included in the search space are associated with decision tree models, as this is one of the most common weak learners that the model uses. However, during our hyperparameter search we encountered some of our highest performing models were using linear functions as base learners. For the final XGBOOST model we selected this was the case.

Table 1: Hyperparameters of Innova-TSN XGBOOST

Parameter	**Value**
colsample_bytree	.6
gamma	4
eta	.5
max_depth	14
min_child_weight	4
subsample	.6

Table 2: Hyperparameters of the models for plugs

Parameter	**2022 Test Set**	**INNOVA Test Set**
booster	gblinear	gblinear
lambda	9.901	.001
alpha	7.136	1.816
subsample	.908	.993
colsample_bytree	.701	.894
colsample_bylevel	.777	.864
colsample_node	.324	.637
max_depth	-	-
min_child_weight	-	-
eta	-	-
gamma	-	-
grow_policy	-	-
max_leaves	-	-

Table 3: Hyperparameters of the models for unplugs

Parameter	**2022 Test Set**	**INNOVA Test Set**
booster	gblinear	gblinear
lambda	.077	8.746e-7
alpha	4.215	.018
subsample	.4406	.533
colsample_bytree	.769	.909
colsample_bylevel	.786	.864
colsample_node	.789	.722
max_depth	-	-
min_child_weight	-	-
eta	-	-
gamma	-	-
grow_policy	-	-
max_leaves	-	-

We also used Optuna to do a hyperparameter search over GCNN hyperparameters, but due to time constraints and computational constraints, we could not do a very intense exploration. Some parameters we tried to explore were: the optimizer to use while training, the learning rate and the number of hours to use in the input (with the intention of maybe using 2 previous weeks to predict over the next one). At the end, we only got to 20 trials, which mostly changed the learning rate only. The final results are described in table 4, but only the learning rate will be used as it is the only parameter that has been explored.

Table 4: Hyperparameters for GCNN over plugs target

Parameter	**value**
Optimizer	RMSprop
LR	0.000818
number of input hours	168

4 Results

4.1 Metrics

We evalutated the performance of our models using 3 metrics, the Root Mean Squared Error, the Mean Abosolute Error, and the R^2. We chose these since our task is a regression problem. We seek to minimize the Root Mean Squared Error and the Mean Absolute error, and maximize the R^2. All of our models performed better than our Baseline model in this regard. The XGBOOST model had the best results from all the models across all

Table 7: MAE comparison for the Innova-TSN test set

Model	**Plugs**	**Unplugs**
Innova-TSN XGBOOST	1.106	1.106
Our XGBOOST	2.362	2.371
GCNN without lagged weather	2.844	2.84

metrics and test sets. The metrics of the models' performance on the 2022 test set can be seen in Tables 5 and 6. We can also see that adding lagged weather variables to the GCNN improved performance of the model across all metrics albeit only slightly.

Table 5: Metrics of the models for plugs on the 2022 test set

Model	**RMSE**	**MAE**	R^2
Baseline	2.831	1.573	.077
XGBOOST	2.053	1.395	.331
GCNN with lagged weather	2.326	1.541	.2
GCNN without lagged weather	2.364	1.584	.168

Table 6: Metrics of the models for unplugs on the 2022 test set

Model	**RMSE**	**MAE**	R^2
Basic Baseline	2.831	1.573	.077
XGBOOST	2.219	1.513	.273
GCNN with lagged weather	2.464	1.635	.151
GCNN without lagged weather	2.46	1.647	.147

We were unable to get the .pkl file of the Innova-TSN's XGBOOST therefore it is impossible to distinguish which model has better performance since we cannot use their model on our data. However, their MAE is smaller than ours for a similar time period as seen in table 7. We must also take into account that our data is different than theirs, since we have more features.

4.2 Feature importances

In this section we display the feature importances of our XGBOOST models for the 2022 test set. Since these models were using linear models as base learners the graphs can be interpreted as the coefficients of the linear function. As expected, using the value of our target variable for the same time period for the week before had the strongest influence on our models. For both models, the value of the opposite target variable (e.g. number of unplugs when predicting the number plugs) had the second most amount of influence on the model.

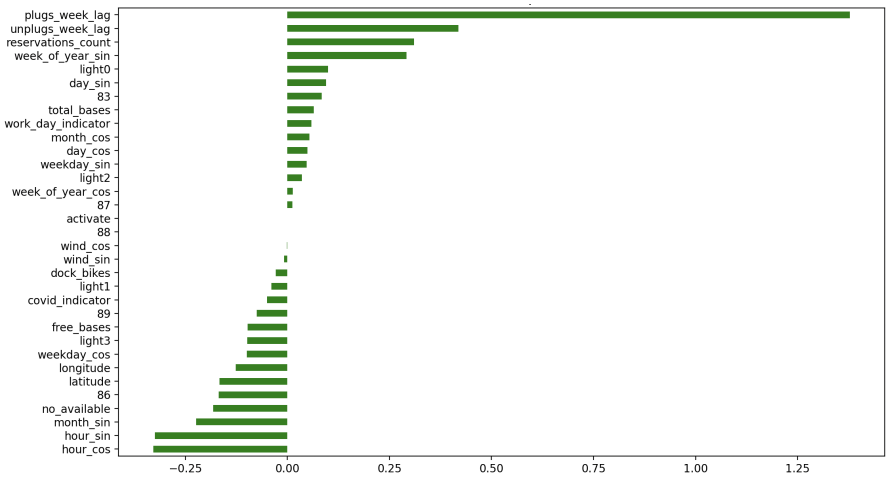


Figure 1: Feature importances for the XGBOOST plug model

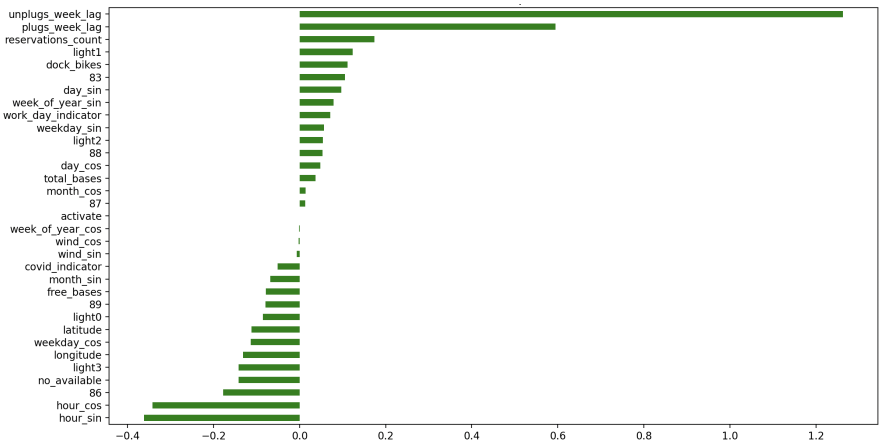


Figure 2: Feature importances for the XGBOOST unplug model

There are ways to get model explainability from a neural network but we were unable to obtain any measures of this for our GCNN due to time constraints.

4.3 Comparisons INNOVA test set

Visualizations for our predictions can be seen in the Appendix starting with Figure 5. These are the predictions from June 20th, 2021 to June 27th, 2021 (the Innova-TSN Test set) for all of the stations. The first thing to notice is that the GCNN under predicts the number of plugs and unplugs while the XGBOOST model over predicts them. Moreover, the predictions of the GCNN are smoother than those of the XGBOOST. Smoother in the sense that they capture the same demand flow throughout the days, but the changes in demand hour by hour are less pronounced.

In subsequent figures we see these predictions at more granular levels. Specifically we filter these predictions

for four geographic areas: Azca, Cuzco, Chamberí, and El Retiro. These are the areas we picked to represent the following city districts respectively: office, shopping, residential, and leisure.

In Azca we expect the amount of plugs and unplugs to be less during the weekend and more during the weekdays. Moreover, we expect the number of plugs to be higher than unplugs in the morning hours and vice versa in the afternoon. This pattern would capture the city's commute from the home to work and back throughout the day. From Figure **??** we can see that the GCNN is good at predicting afternoon unplugs from Azca but falls short in predicting the weekday morning plug rush. From Figure 6b we can see the XGBOOST model does a better job of predicting the morning and afternoon rush hour but does a worse job at predicting the demand during the weekend. It is also interesting to see that the demand for bikes on Friday is more evenly distributed throughout the day than previous workdays. The GCNN does a great job predicting the city's literal "unplug" from work on Fridays.

In Cuzco we expect demand to more sporadic since the schedule people run shopping errands does not have a clear trend. Nonetheless, we see in the data 7 that this district is very active in the morning later on in the week, with the most activity happening on Wednesday. With this district our model did not do so good. The GCNN consistently under predicts while the XGBOOST over predicts periods with low demand and under predicts periods with higher demand.

In Chamberí the number of plugs and unplugs is more even through the day but with higher amounts in the afternoon. Perhaps this is because many people work from home in this neighborhood and leave their house afterwards. In this breakout the GCNN **??** does a better job than the XGBOOST. The XGBOOST overshoots the prediction by a large margin.

El Retiro is one of Madrid's most popular parks. We can see that the number of plugs and unplugs is fairly consistent across all of the days of the week. In this breakout the XGBOOST 9b does a better job than the GCNN. The GCNN consistently under predicts the demand while the XGBOOST is consistently very close to the true values.

4.4 Comparisons 2022 test set

Same as before, in the Appendix we can find all the visualizations regarding the results over the 2022 test set. The visualizations include a visualization of the plugs and unplugs over all 2022, where the data is aggregated as the average total number of plugs and unplugs in a day of each week, and also specific visualizations that cover a week and have the following characteristics, they focus on interesting areas as seen in Figure 4; (residential areas: Chamberí, office areas: Azca area and Cuzco Square, and park areas: El Retiro), changes in weather conditions (winter, spring, summer and rain), and holidays (The National Day of Spain, 12 of October). For all visualizations we can see the models for the XGBOOST and GCNN without lagged weather, and for the visualizations that cover the weather changes we can also see the GCNN with lagged weather to see the effect of lagging this variable. Except of the plot of all 2022, the others just represent one week where the data is aggregated as the mean between all the stations that are represented in each plot. For example, for the plots talking about specific areas, only the stations inside those ares are taken into account, while for the rest of visualizations (temperature and holidays), we will use all stations. It is important to notice that true values

between XGBOOST and GCNN plots may differ, as for the GCNN we had to remove stations from the dataset to have a static graph.

Analyzing all 2022

In Figure 10, we can already see some differences between XGBOOST and GCNN, where interestingly GCNN captures a lot better the trends and seasonalities of the data, but has problems capturing some outliers, especially the positive ones. It is interesting how even though XGBOOST has a lower RMSE it fails to capture the overall behavior of the data.

Analyzing specific areas

The first thing to notice in these plots is that we can see how in the office areas (Azca area, Figure 11 and Cuzco Square, Figure 12) there is a pattern. We can see how plugs are a lot higher than unplugs at 9am in the morning, which represents people arriving to work, and unplugs are a lot higher than plugs at 6pm, which represents people leaving to go home. Also, we can see how the first and last day (Sunday and Saturday respectively), have much lower activity and do not contain any pattern. Then, in Chamberí, Figure 13 we can see the inverse behavior, where unplugs are higher in the morning and plugs are higher in the evening, even though it is not that noticeable. Finally, El Retiro, Figure14 shows a more balanced behavior where there is not any specific pattern other than the fact that most movements are done at 9am, 6pm, and somewhere around 15pm. In terms of the predictions of the models, we can see how they both predict a very similar pattern over all working days and weekend days separately. We can see how GCNN slightly understands the effect of the peak hours but falls short to get the morning peaks and always predicts higher peaks for unplugs, which means it is not understanding the behavior on the residential area. On the other hand, we can see how XGBOOST adapts better to each scenario, predicting higher plugs values in the morning and higher unplugs values in the evening in both Cuzco Square and Azca's area, and also balancing better the predictions over peak hours for Chamberí and El Retiro.

Analyzing weather effect

The figures 15, 16 17 and 18 represent weeks of winter, spring, summer and a rainy week respectively. Here we have also added a GCNN model trained with lagged weather variables, which means it is using a 100% accurate forecast of weather as features for prediction. In the winter graph, the first thing we can notice is that there is a period of 2 days with very low temperatures of around -10 C, which affects the true values decreasing the number of both plugs and unplugs. We can see how all models correctly detect this effect, which means the models understand the effect the temperature is having on predictions. Then, in the spring plot, both models overestimate the predictions, and in the summer plot, all models capture the true value. Interestingly, in the rainy plot, there is a lot of rain that falls in day hours in the 5th day, and neither of the models is able to capture the effect, as they both predict a normal day outcome. So with the analysis over all these plots, we can say that lagging the weather variables has no effect on the predictions.

Analyzing holidays effect

Finally, in figure 19, we can see how the 4th day , which represents a Thursday and also is 12th of October (National Day of Spain), has a drop in use at least in the morning. The GCNN model is not seeing this effect,

while XGBOOST is partially correcting the predictions to adjust to it.

5 Conclusion and Discussion

In this project we have created a BiciMAD bikesharing system dataset using several data sources such as EMT and DatosAbiertos, which allowed us to build a complete and coherent database with a considerable range of features with high predictive power for plugs and unplugs prediction and includes also hourly weather measures at each bike station. We have also done a preliminary prediction study over the dataset gathered using state-of-the-art techniques such as XGBOOST optimized with Optuna and Graphic Convolutional Neural Networks in a static graph configuration, achieving higher performance for XGBOOST even though in the literature usually GCNN has better performance. We have tested the models in several environments such as areas with different characteristics (office areas, residential areas and parks), or time spans with different weather (winter, spring, summer, and rainy days) and also special events such as holidays in work days, observing how the models react to those environments; in most cases capturing the effects present in each test scenario.

5.1 Future lines of work

Even though we have obtained very interesting results and we have correctly worked through the problems in the data sources to get the final BiciMAD dataframe, this project is far from finished. Some future lines of work are presented here:

- Realize a deep quality data assessment. This project has been done with a very strict time constraint, which meant that only necessary data quality assessment has been done to the process, but it is far from being enough.
- Increment the dataframe with more variables. More variables can be added, such as connections with other means of transport.
- Feature engineering. The current dataframe has a lot of potential for exploitation that has not been used yet.
- GCNN with dynamic graphs. Implementing GCNN approach with a dynamic graph could be very interesting, first of all, it would allow to train this model with all existing data instead of a subset, and second, would make the model robust to changes in the stations network as well as maybe provide insights on how the stations relate with each other.
- GCNN explainability. Exploring in the direction of AI explainability with this model could lead to understanding it better and why it is not performing as well as literature shows.

All the code used for this project can be found at [6].

References

[1] Vitória Albuquerque, Miguel Sales Dias, and Fernando Bacao. "Machine learning approaches to bike-sharing systems: A systematic literature review". In: *ISPRS International Journal of Geo-Information* 10.2 (2021), p. 62.

[2] Jiandong Bai et al. "A3t-gcn: Attention temporal graph convolutional network for traffic forecasting". In: *ISPRS International Journal of Geo-Information* 10.7 (2021), p. 485.

[3] El Diario.es. *El mapa ciudadano de las nuevas estaciones de Bicimad (según las va instalando el Ayuntamiento)*. URL: https://www.eldiario.es/madrid/somos/mapa-ciudadano-nuevas-estaciones-bicimad-instalando_1_9967736.html. (accessed: 04.07.2023).

[4] Elliot Fishman. "Bikeshare: A review of recent literature". In: *Transport reviews* 36.1 (2016), pp. 92–113.

[5] Ruiying Guo et al. "BikeNet: Accurate Bike Demand Prediction Using Graph Neural Networks for Station Rebalancing". In: *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. 2019, pp. 686–693. DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00153.

[6] Roset Javier Gutierrez Erika. *Master Thesis github repository*. URL: https://github.com/javi99/MT_predicting_BSD. (accessed: 04.07.2023).

[7] Timothy L Hamilton and Casey J Wichman. "Bicycle infrastructure and traffic congestion: Evidence from DC's Capital Bikeshare". In: *Journal of Environmental Economics and Management* 87 (2018), pp. 72–93.

[8] Andreas Kaltenbrunner et al. "Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system". In: *Pervasive and Mobile Computing* 6.4 (2010), pp. 455–466.

[9] Zhaoyu Kou et al. "Quantifying greenhouse gas emissions reduction from bike share systems: a model considering real-world trips and transportation mode choice patterns". In: *Resources, Conservation and Recycling* 153 (2020), p. 104534. ISSN: 0921-3449. DOI: https://doi.org/10.1016/j.resconrec.2019.104534. URL: https://www.sciencedirect.com/science/article/pii/S0921344919304409.

[10] Yuebing Liang, Guan Huang, and Zhan Zhao. *Cross-Mode Knowledge Adaptation for Bike Sharing Demand Prediction using Domain-Adversarial Graph Neural Networks*. 2022. arXiv: 2211.08903 [cs.LG].

[11] Lei Lin, Zhengbing He, and Srinivas Peeta. "Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach". In: *Transportation Research Part C: Emerging Technologies* 97 (2018), pp. 258–276. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2018.10.011. URL: https://www.sciencedirect.com/science/article/pii/S0968090X18300974.

[12] Pytorch. *Recurrent Graph Convolutional Layers*. URL: https://pytorch-geometric-temporal.readthedocs.io/en/latest/modules/root.html. (accessed: 04.07.2023).

[13] Benedek Rozemberczki et al. "PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models". In: *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 2021, pp. 4564–4573.

[14] Tae San Kim, Won Kyung Lee, and So Young Sohn. "Correction: Graph convolutional network approach applied to predict hourly bike-sharing demands considering spatial, temporal, and global effects". In: *PLOS ONE* 17.3 (Mar. 2022), pp. 1–1. DOI: 10.1371/journal.pone.0266221. URL: https://doi.org/10.1371/journal.pone.0266221.

[15] Department of Economic United Nations and Population Division Social Affairs. *World Urbanization Prospects: The 2018 Revision*. New York: United Nations: United Nations, 2019.

[16] Carlos M Vallez, Mario Castro, and David Contreras. "Challenges and opportunities in dock-based bike-sharing rebalancing: a systematic review". In: *Sustainability* 13.4 (2021), p. 1829.

[17] Yongping Zhang and Zhifu Mi. "Environmental benefits of bike sharing: A big data-based analysis". In: *Applied Energy* 220 (2018), pp. 296–301. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2018.03.101. URL: https://www.sciencedirect.com/science/article/pii/S0306261918304392.

6 Appendix

6.1 Weather stations assignation to bike stations

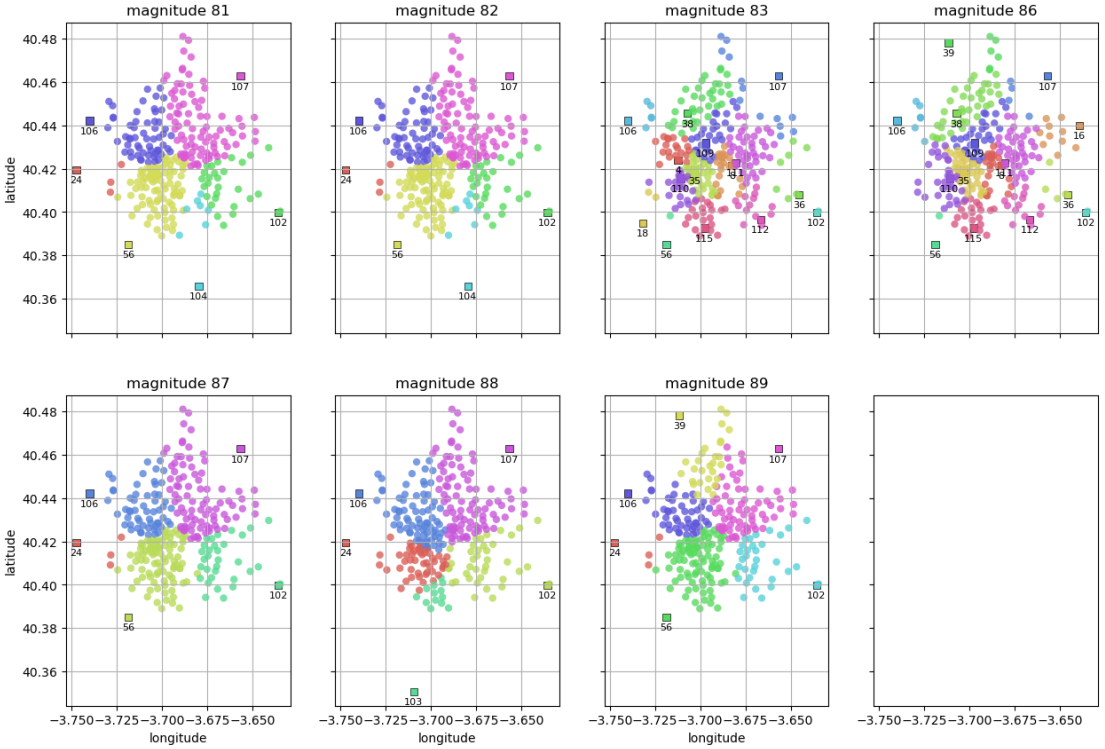


Figure 3: Relation between bike stations and weather stations for may 2021

6.2 Stations visualization

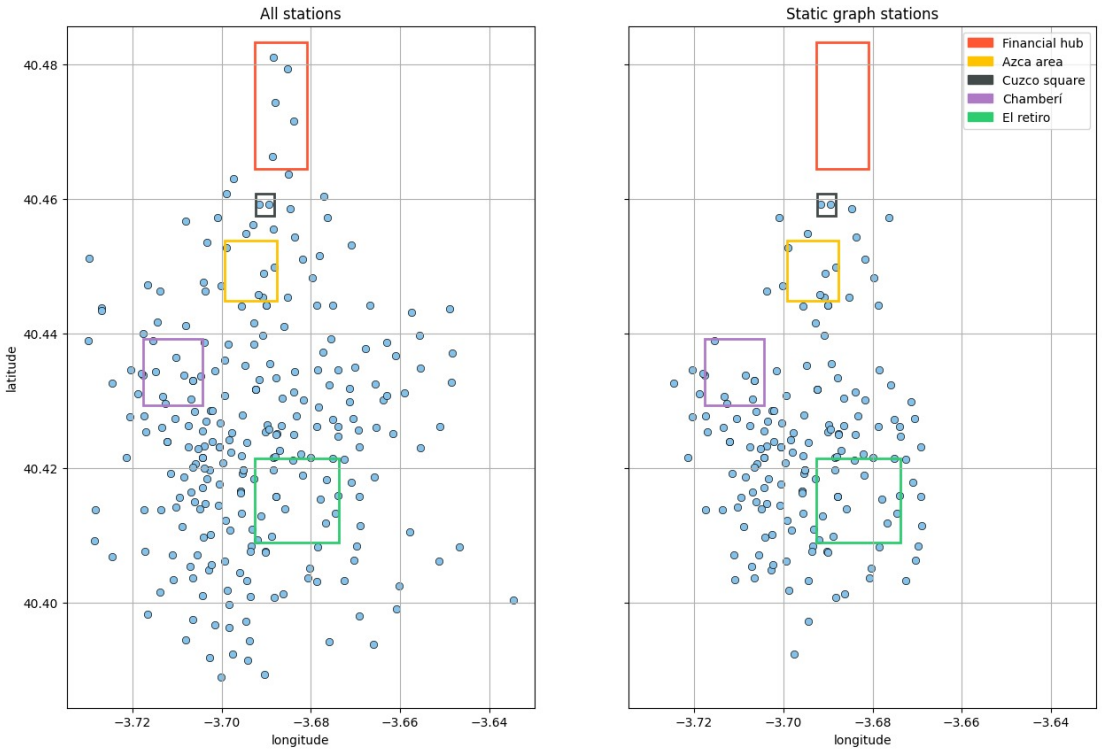
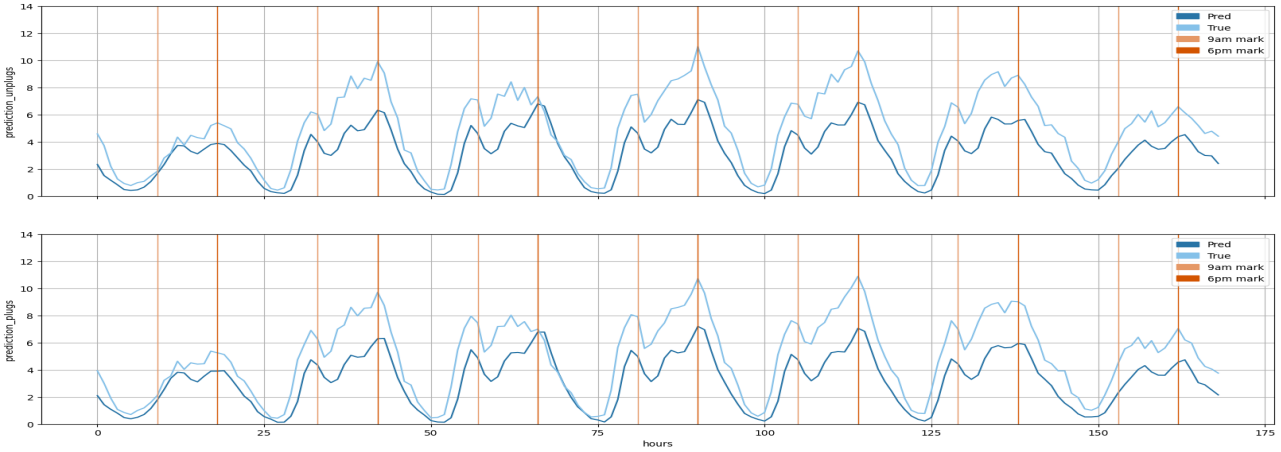


Figure 4: Stations with specific areas for analysis

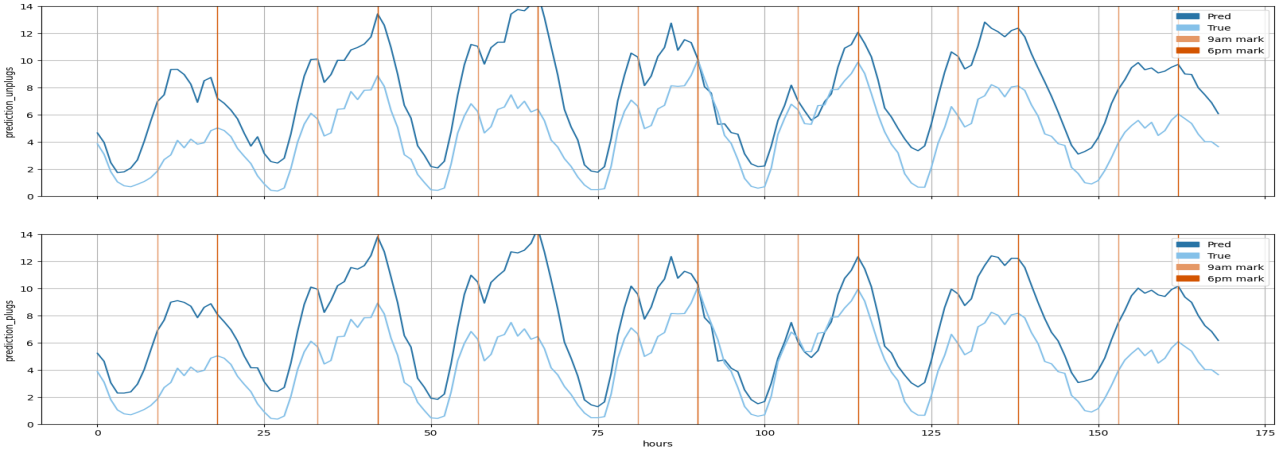
6.3 Visualizations over INNOVA test set

The week that is shown in the INNOVA test set plots 20-06-2022 to 27-06-2022.

6.3.1 Analyzing all stations



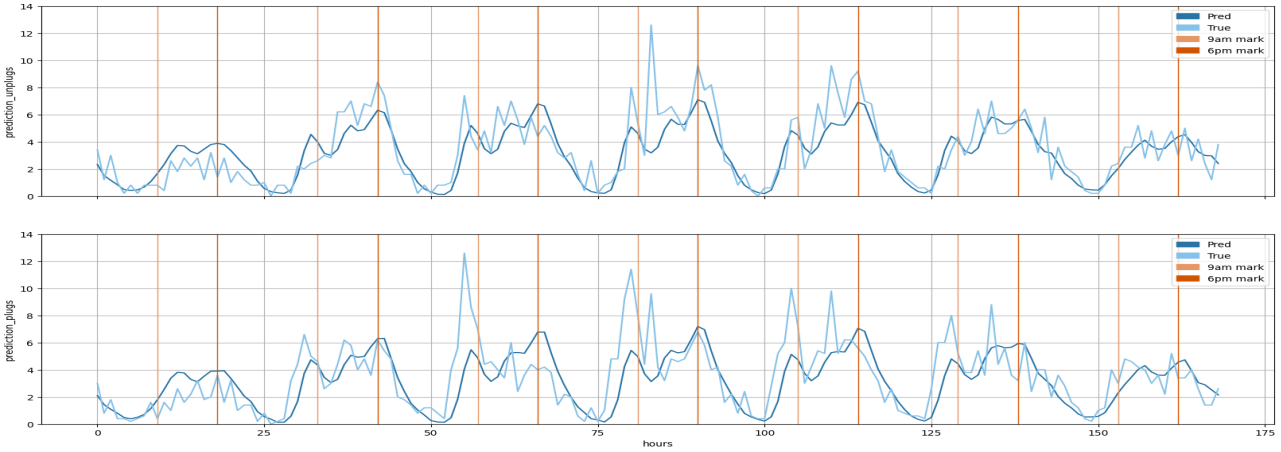
(a) Predictions of GNN



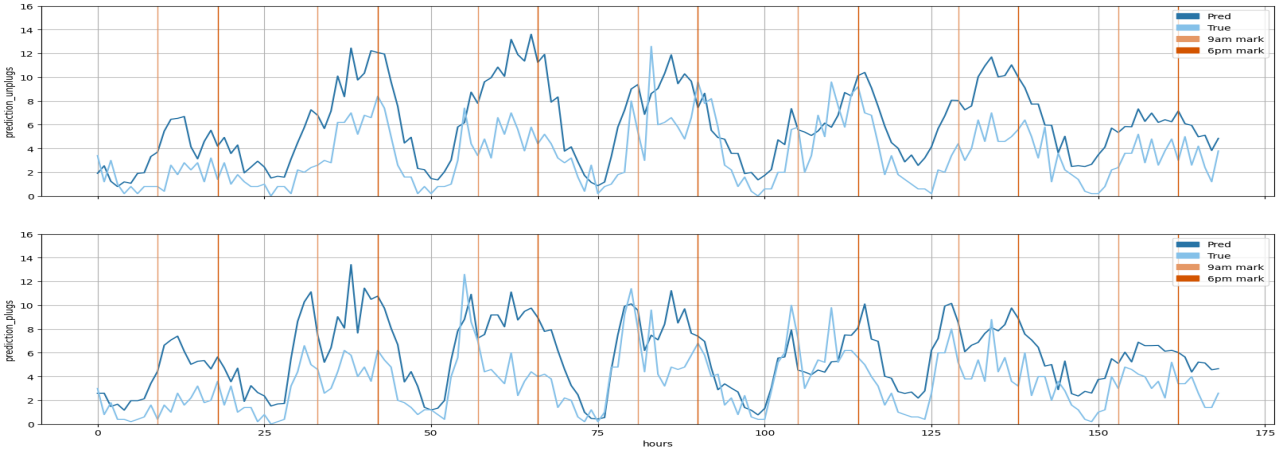
(b) Predictions of XGBoost

Figure 5: Predictions over INNOVA's test set

6.3.2 Analyzing specific areas

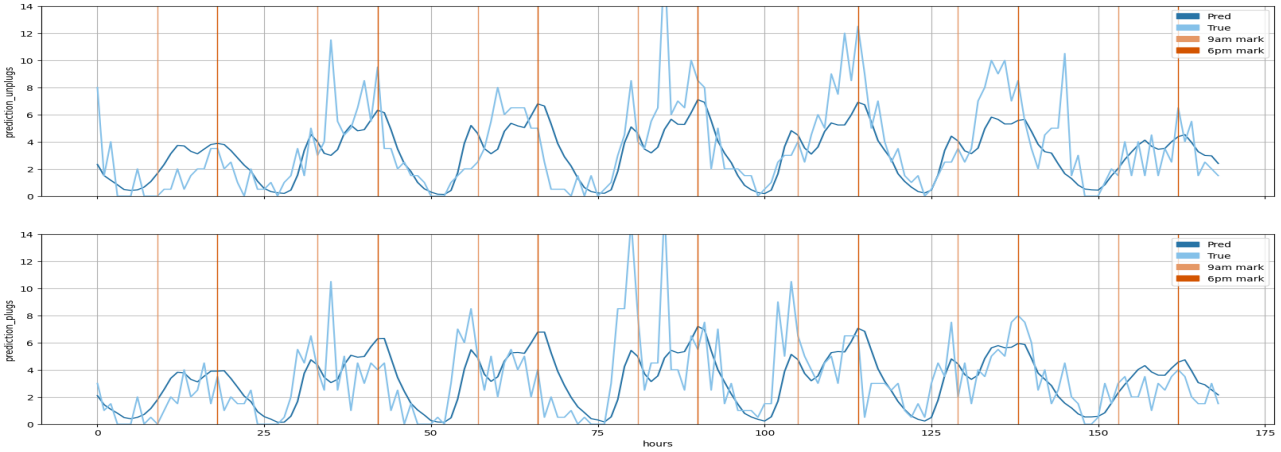


(a) Predictions of GNN

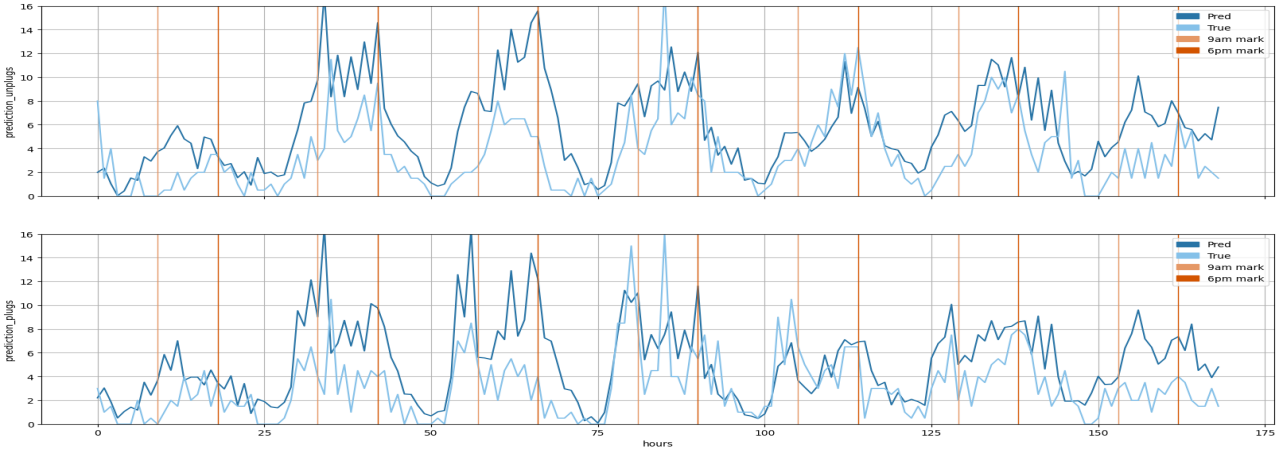


(b) Predictions of XGBoost

Figure 6: Predictions over INNOVA's test set in Azca's area

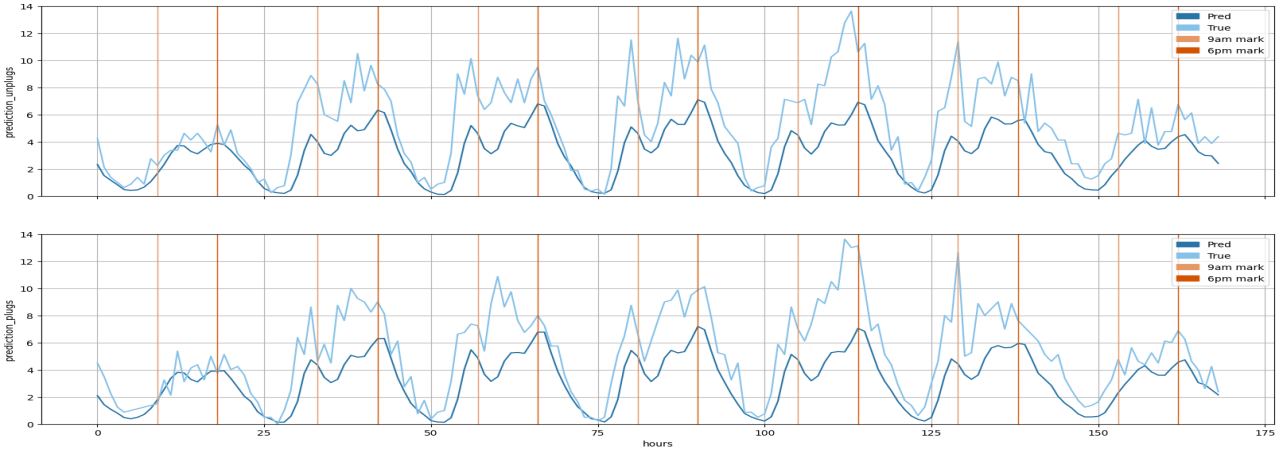


(a) Predictions of GNN

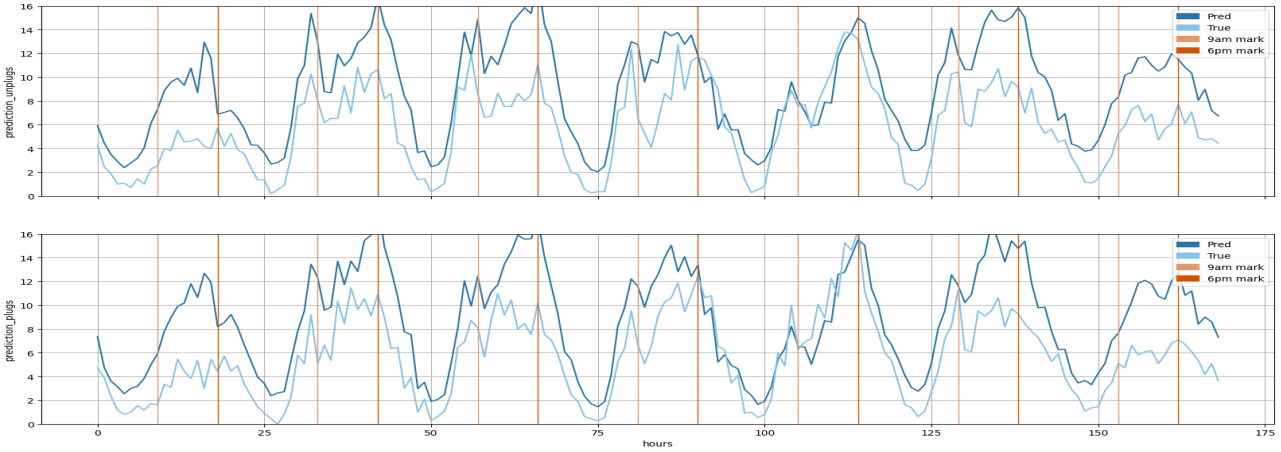


(b) Predictions of XGBoost

Figure 7: Predictions over INNOVA's test set in Cuzco square area

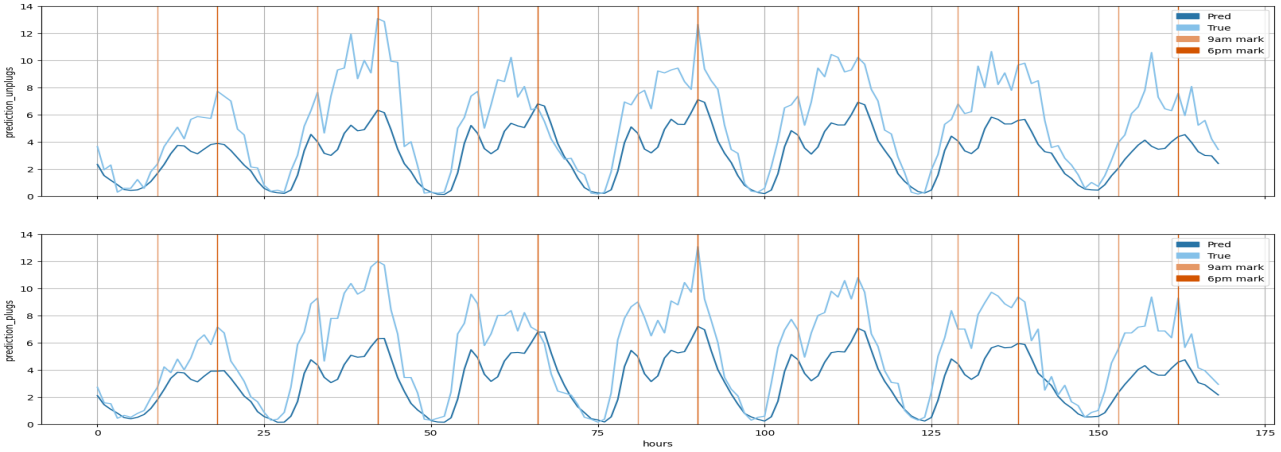


(a) Predictions of GNN

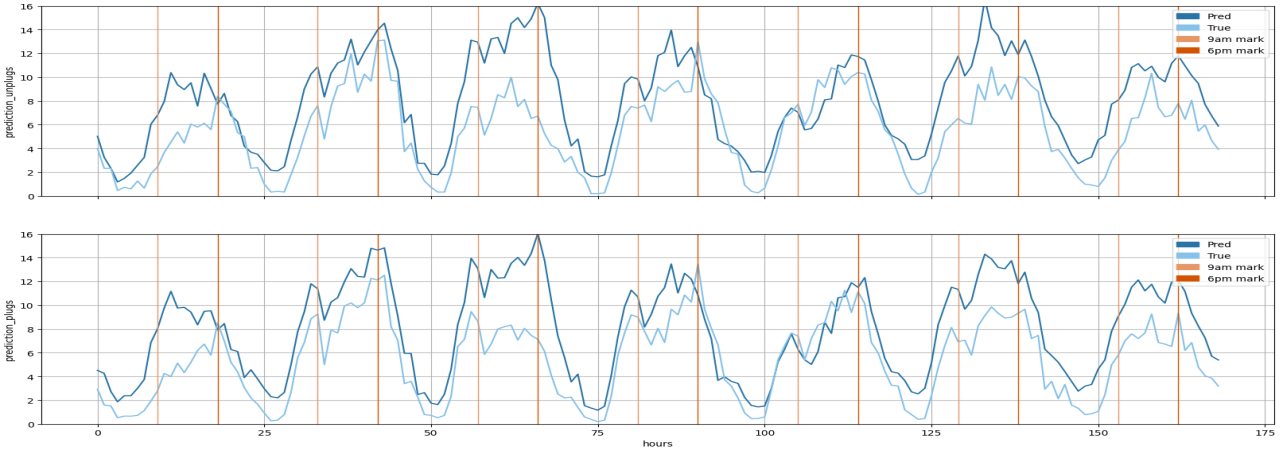


(b) Predictions of XGBoost

Figure 8: Predictions over INNOVA's test set in Chamberí area



(a) Predictions of GNN



(b) Predictions of XGBoost

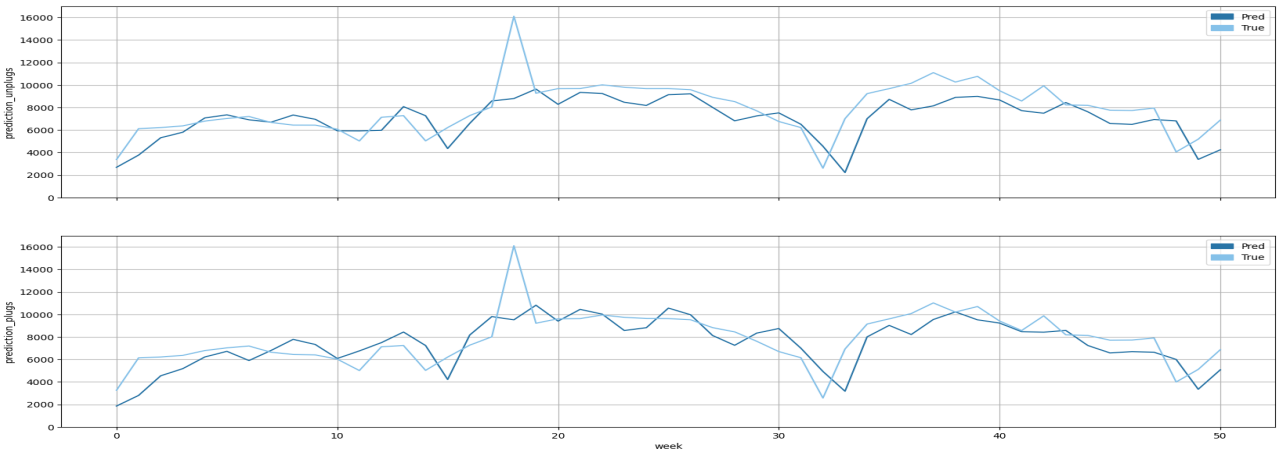
Figure 9: Predictions over INNOVA's test set in El Retiro

6.4 Visualization over all 2022 test set

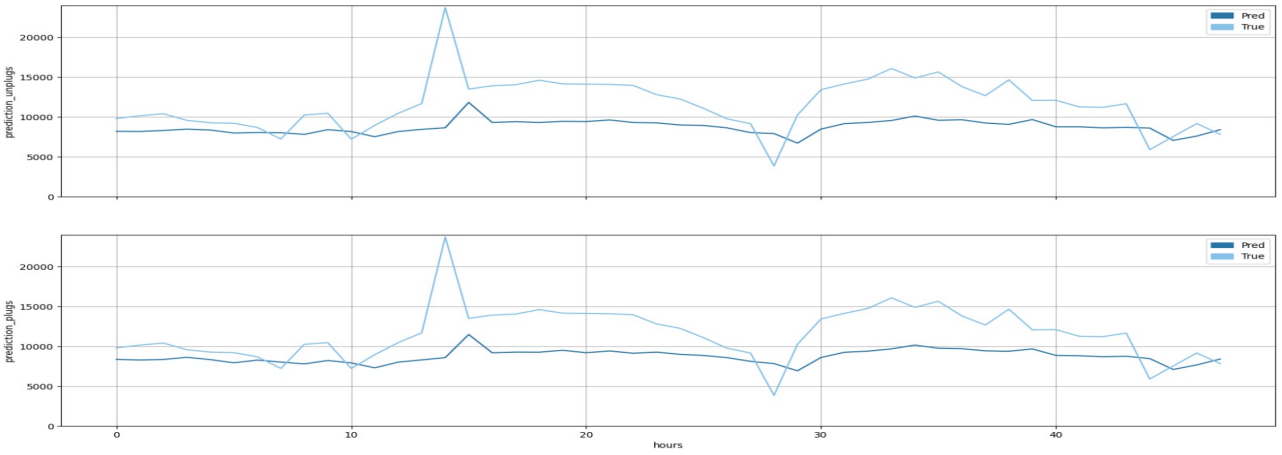
For the 2022 test set, the time periods chosen for each analysis are the following:

- Analyzing all stations: from 02-01-2022 to 31-12-2022
- Analyzing specific areas: from 12-06-2022 to 19-06-2022
- Analyzing weather effect, winter: from 10-02-2022 to 17-02-2022
- Analyzing weather effect, spring: from 12-04-2022 to 19-04-2022
- Analyzing weather effect, summer: from 13-07-2022 to 20-07-2022
- Analyzing holidays effect, Hispanity day (October the 12th): from 09-10-2022 to 16-10-2022

6.4.1 Analyzing all stations



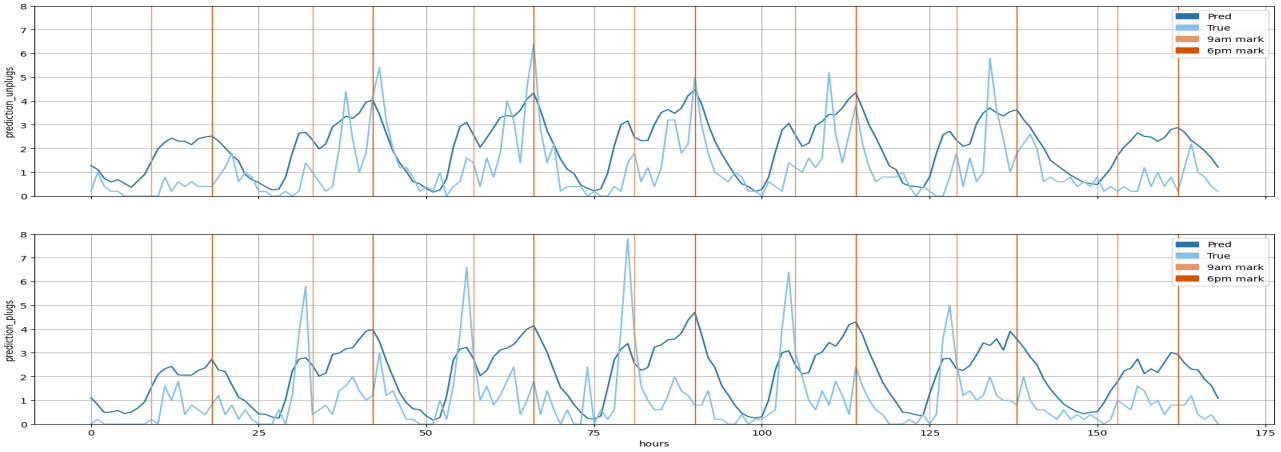
(a) Predictions of GNN



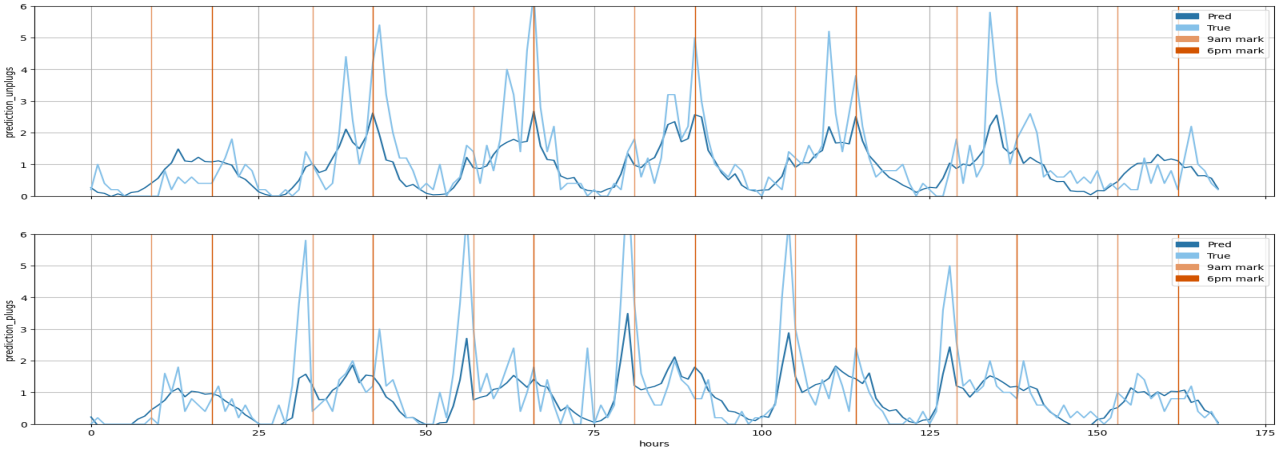
(b) Predictions of XGBoost

Figure 10: Predictions over 2022 test set

6.4.2 Analyzing specific areas

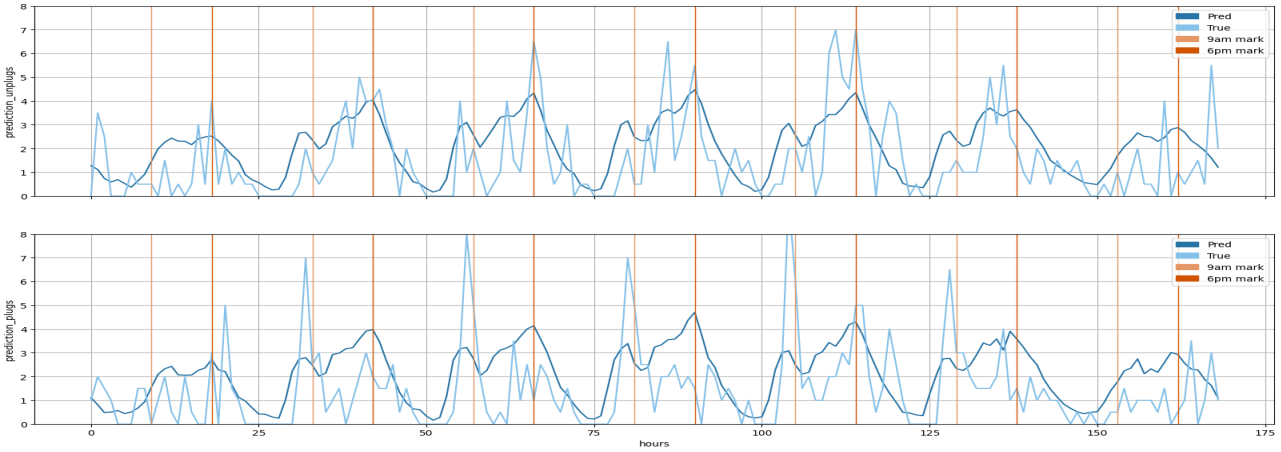


(a) Predictions of GNN

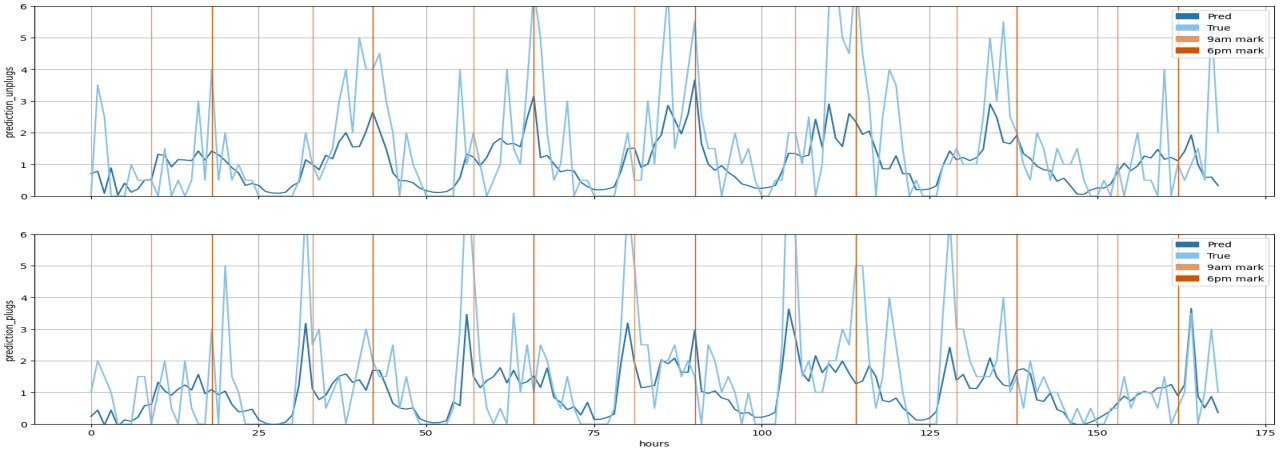


(b) Predictions of XGBoost

Figure 11: Predictions over 2022 test set in Azca's area

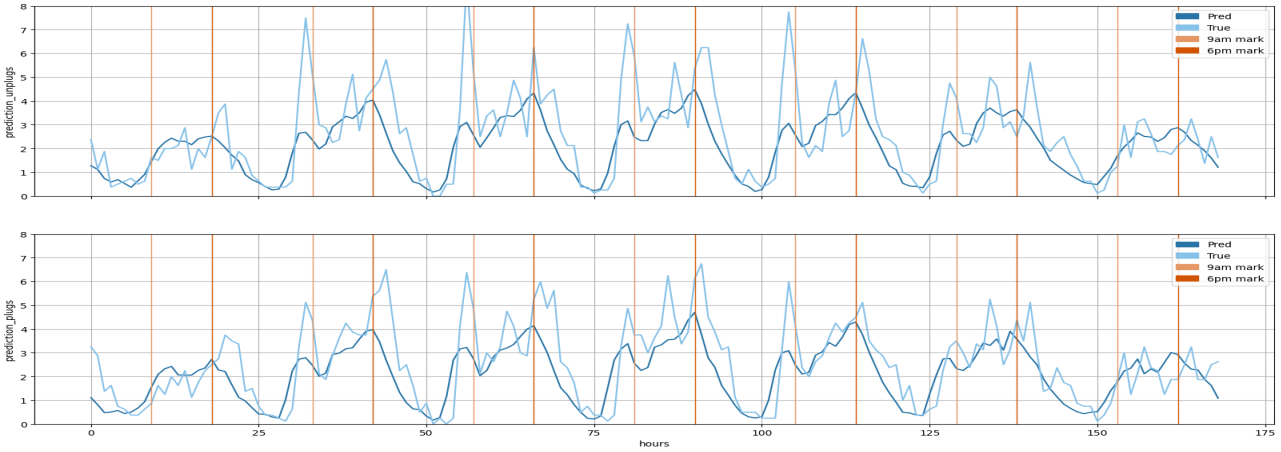


(a) Predictions of GNN

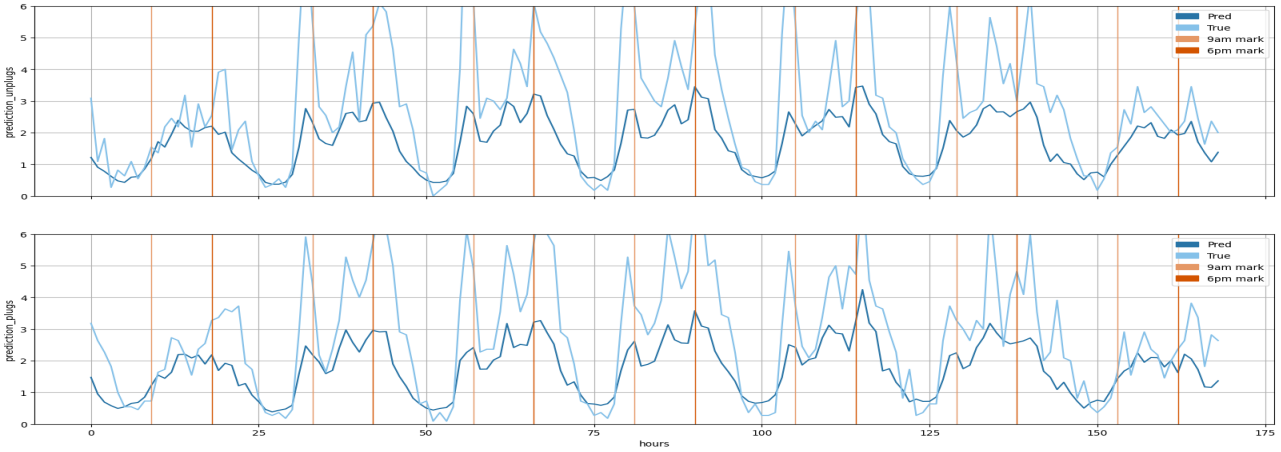


(b) Predictions of XGBoost

Figure 12: Predictions over 2022 test set in Cuzco square

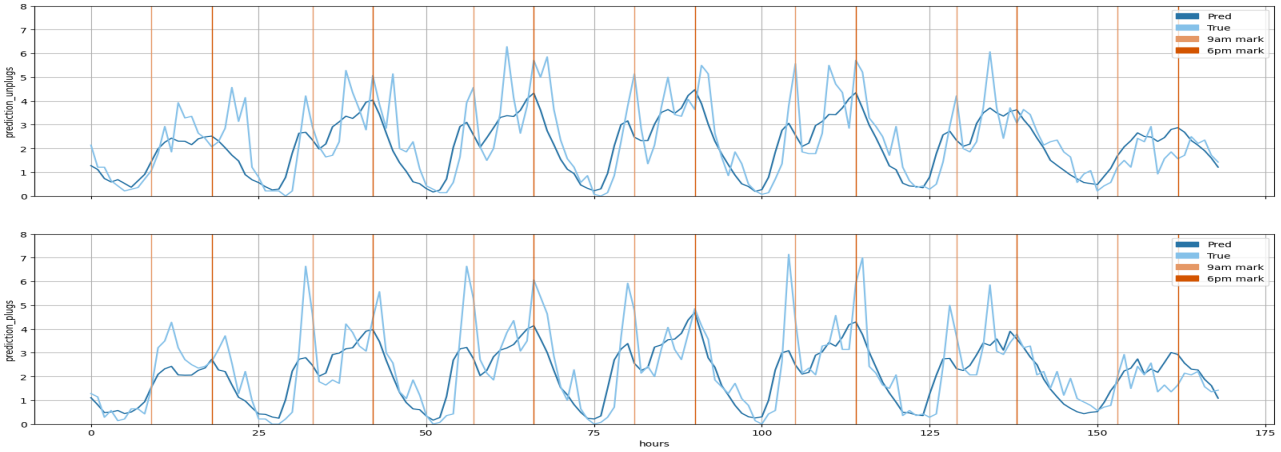


(a) Predictions of GNN

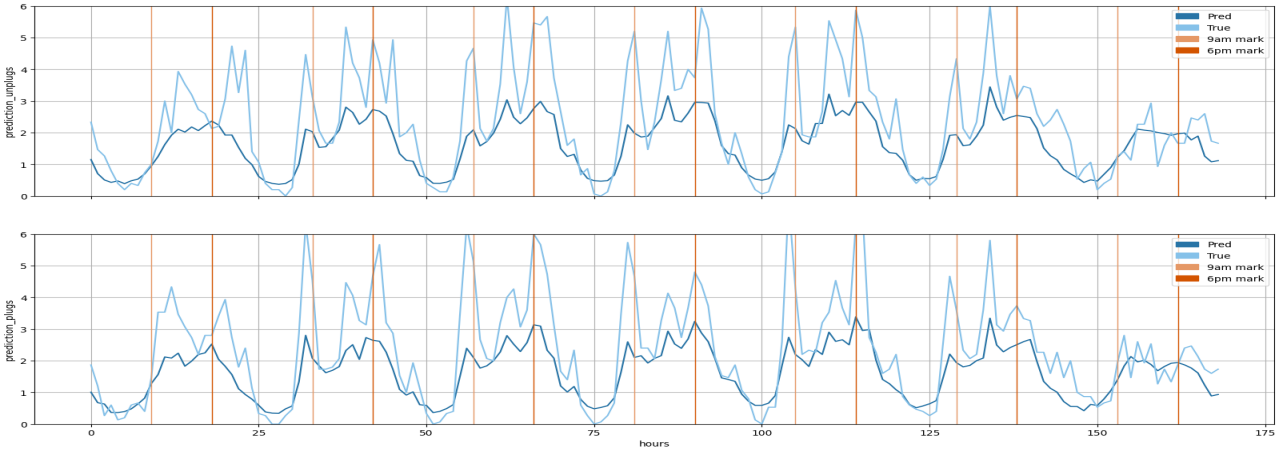


(b) Predictions of XGBoost

Figure 13: Predictions over 2022 test set in Chamberí



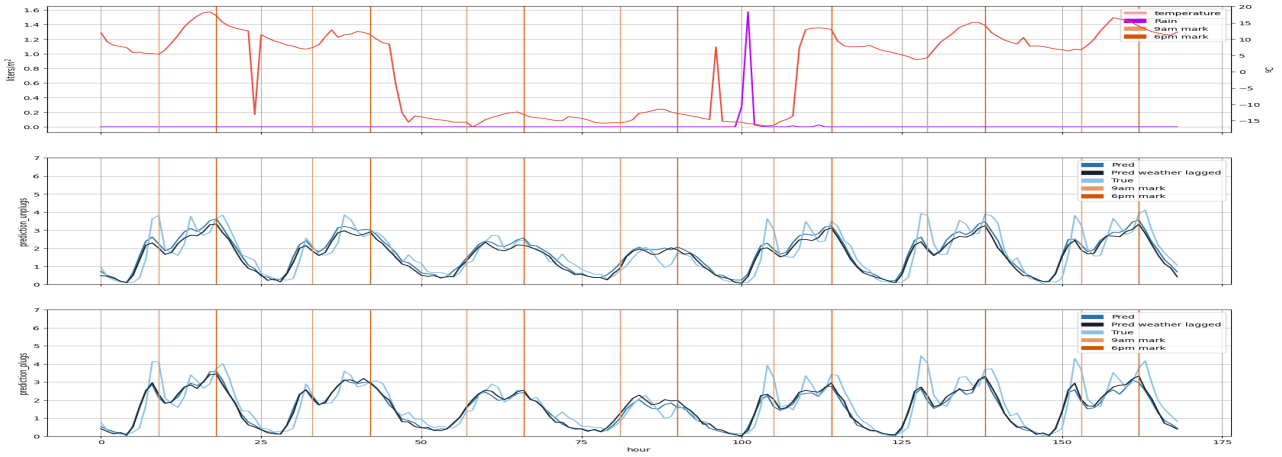
(a) Predictions of GNN



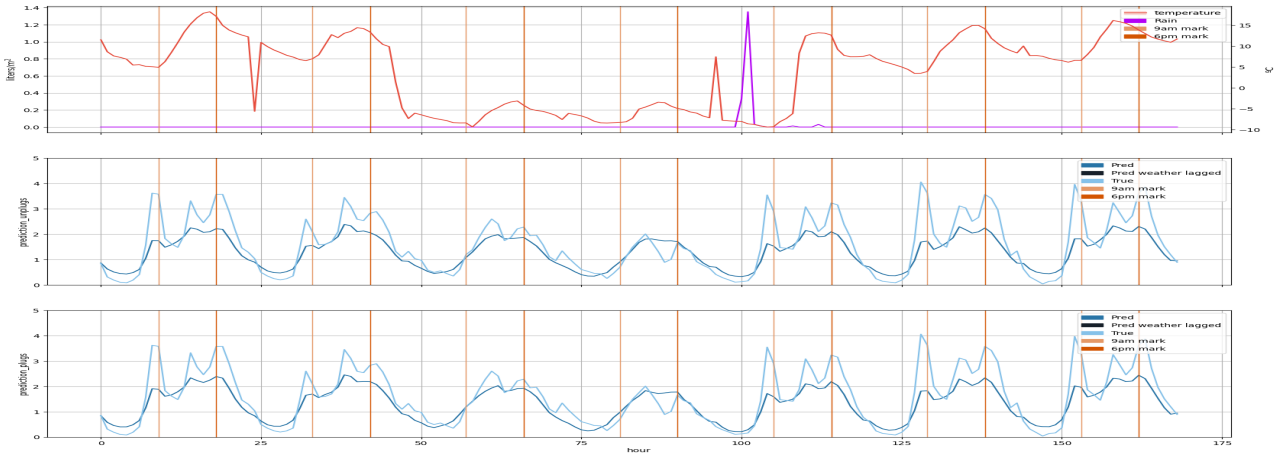
(b) Predictions of XGBoost

Figure 14: Predictions over 2022 test set in El Retiro

6.4.3 Analyzing weather effect

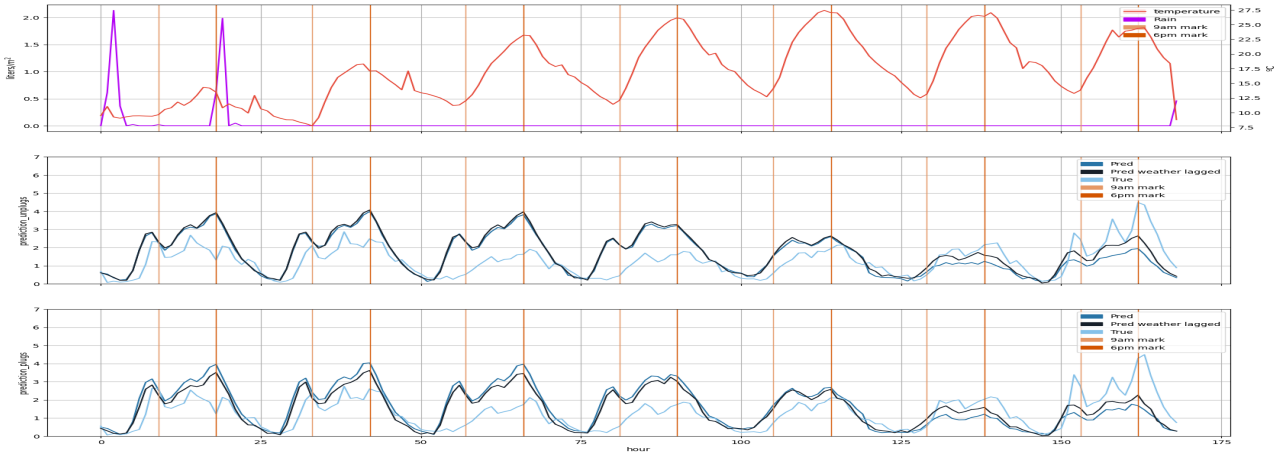


(a) Predictions of GNN

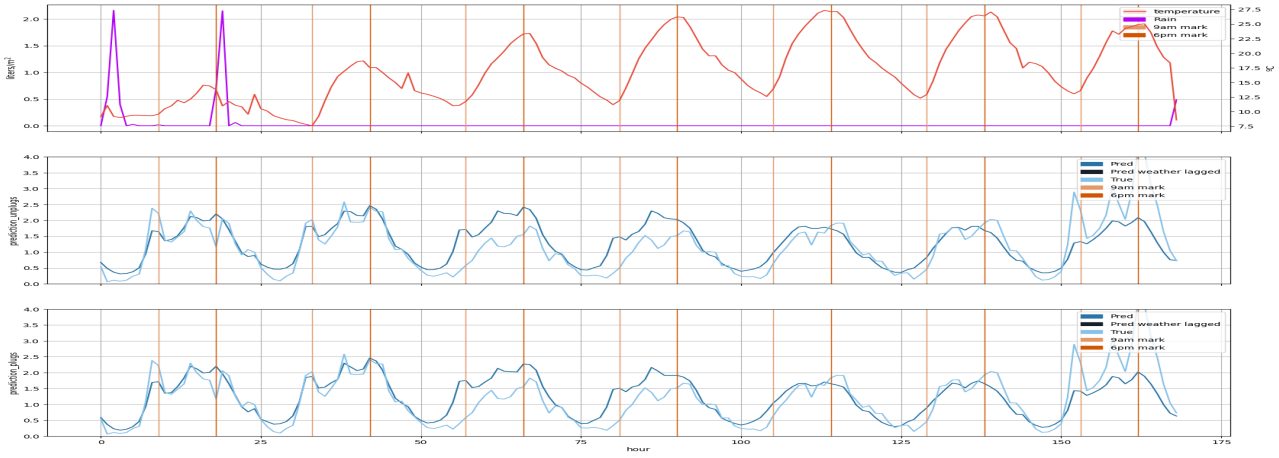


(b) Predictions of XGBoost

Figure 15: Predictions over 2022 test set in a winter week

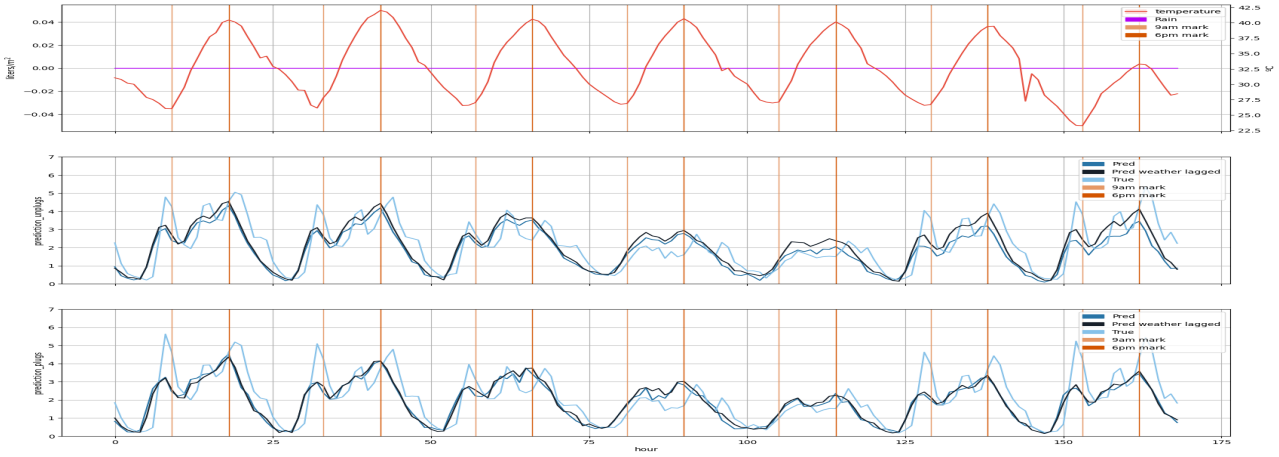


(a) Predictions of GNN

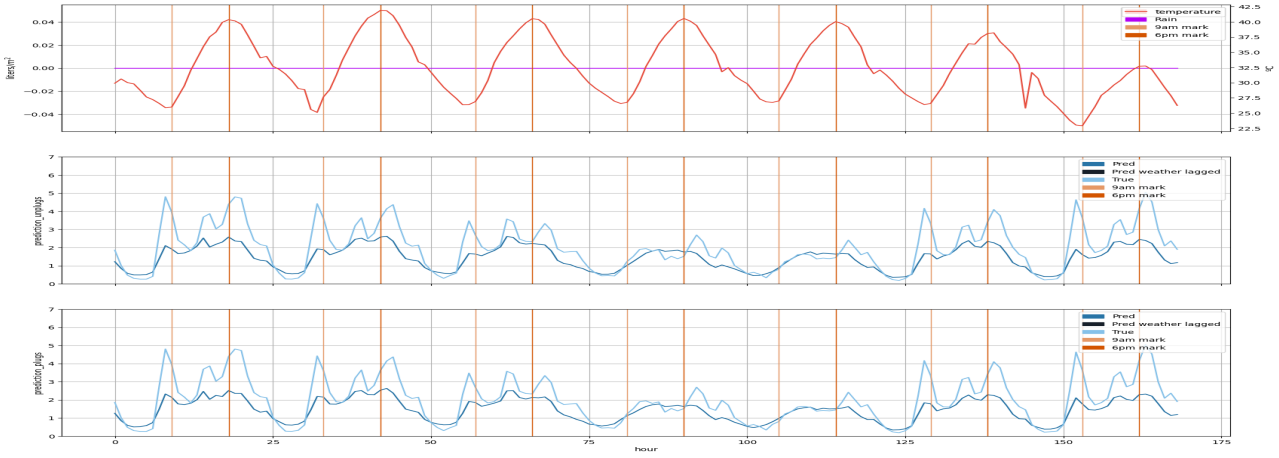


(b) Predictions of XGBoost

Figure 16: Predictions over 2022 test set in a spring week

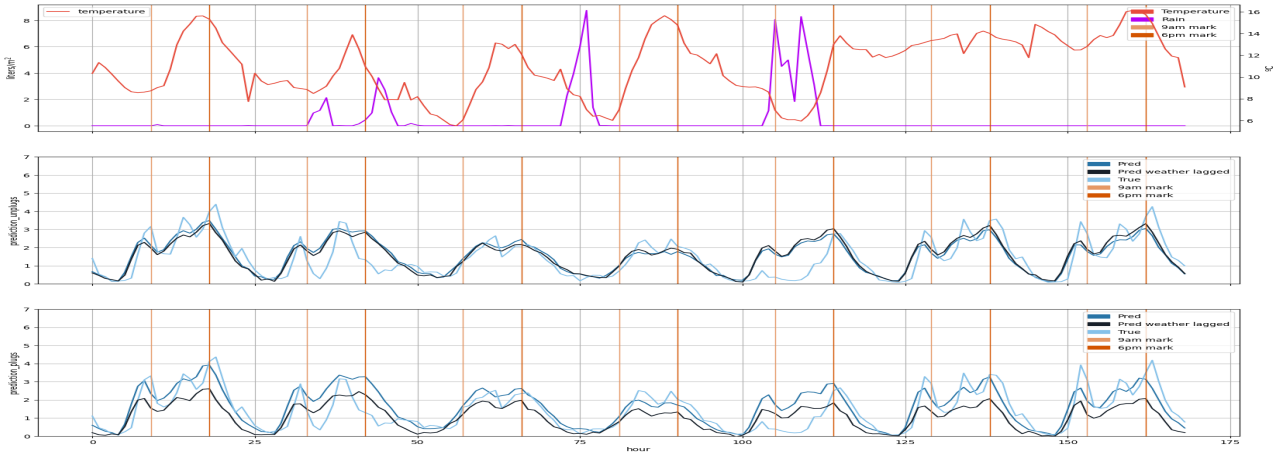


(a) Predictions of GNN

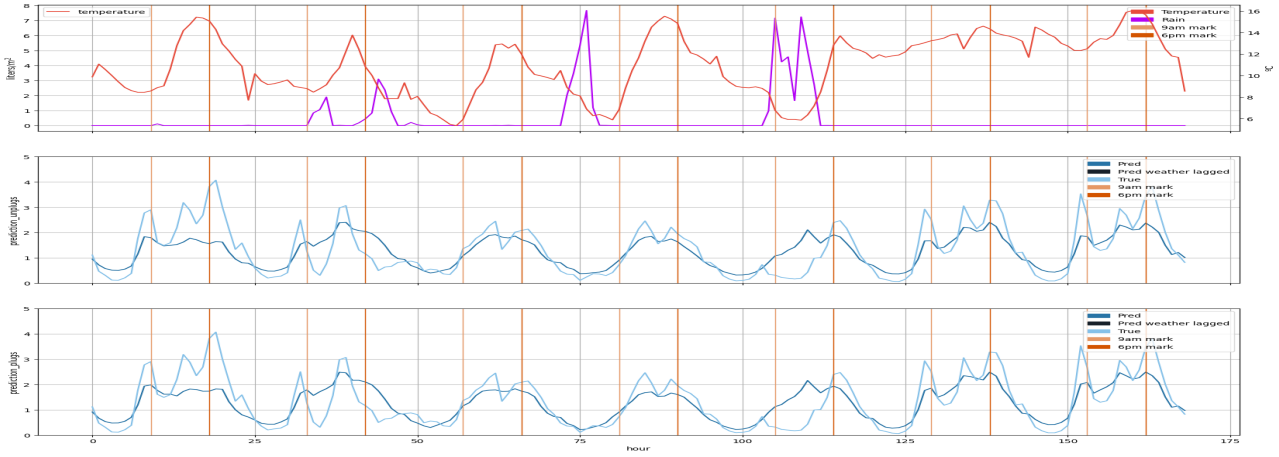


(b) Predictions of XGBoost

Figure 17: Predictions over 2022 test set in a summer week



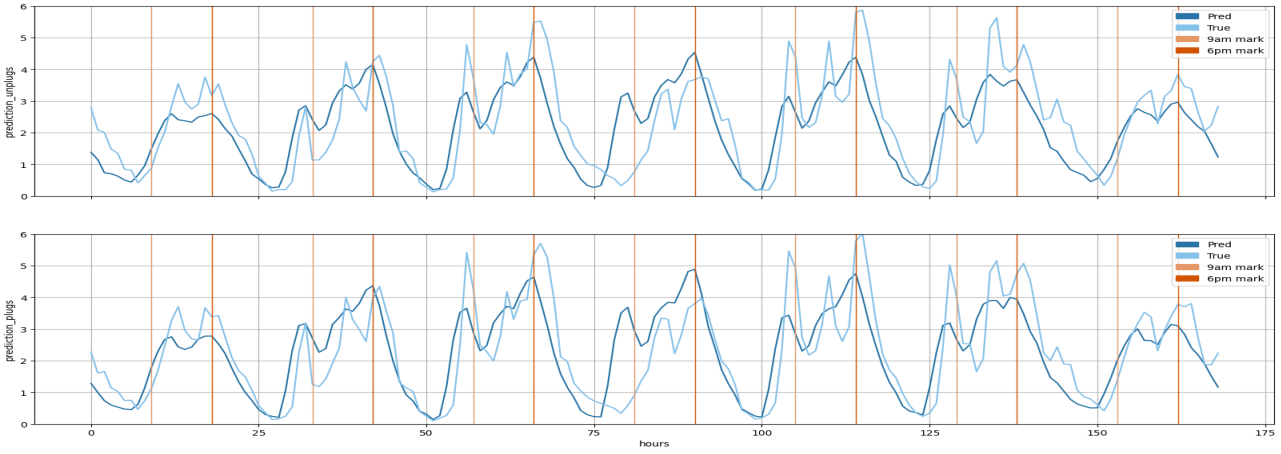
(a) Predictions of GNN



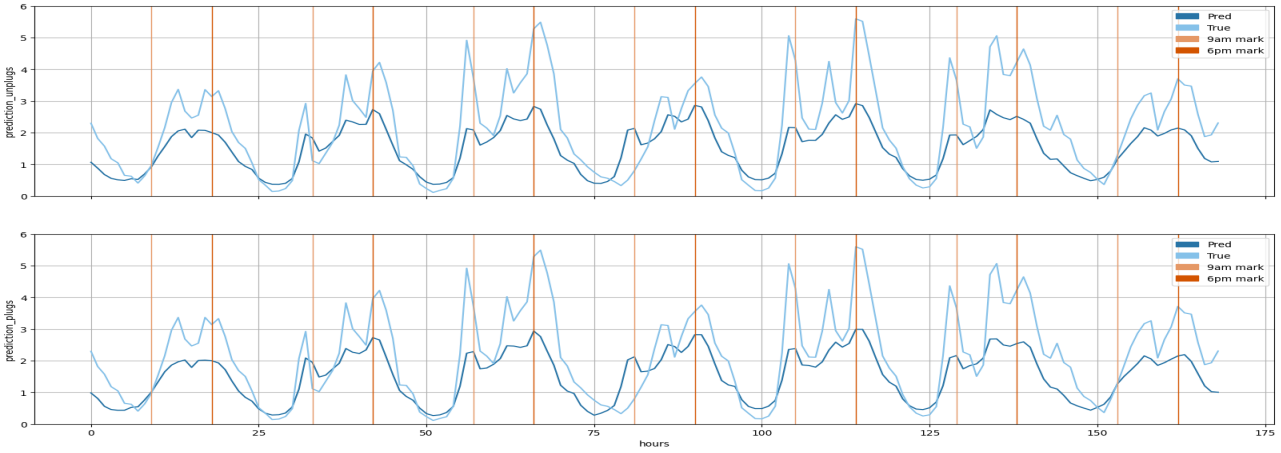
(b) Predictions of XGBoost

Figure 18: Predictions over 2022 test set in a rainy week

6.4.4 Analyzing holidays effect



(a) Predictions of GNN



(b) Predictions of XGBoost

Figure 19: Predictions over 2022 test set in National Day of Spain week (4th day in the represented week)

6.5 Data Errors

Unfortunately, most of our time during the semester was devoted to data cleaning tasks. Below we outline a couple of the errors we had to deal with.

6.5.1 Webscraping

The first problem we encountered with the datasets that we were referred to was that they only had data through 2021 but we wanted to include 2022 for our study. We had to email and call EMT Madrid multiple times to upload this data to their website. When the data was eventually uploaded, some of the files were json and some of them were csvs. The json files were in a format not readable with the typical functions of python's

json package so we had to develop an algorithm to extract the information from the file and then structure it as a json. Furthermore, the columns and data types in these files were inconsistent. We wrote code to account for this and as we were making the final dataframe it was also encountered that these datasets had duplicates and missing hours, which we also accounted for. During this process we realized that the csv file for October 2021 trip transactions was so corrupt that it was unusable. We again reached out to EMT Madrid to get a clean file but they were not responsive. Therefore, we imputed these values for October 2021 in a linear fashion.

6.5.2 Bike stations coordinates inconsistencies

When processing the bike stations dataframe, we encountered an inconsistency with regard to the coordinates of these. We discovered that some stations were registered with different coordinates each time, which represented a risk of losing coherence in the dataframe for future steps where we used coordinates to relate bike stations with weather stations for example. To study this problem, we created figure 20, where we first measure the maximum distance between the coordinates that each station has associated, and then count the number of stations inside the problematics one that represents the same distance. The intention was to see if this is due to some missing decimals, or other errors. We discovered that in most cases the discrepancies were due to missing decimals, or coordinate registers that were separated less than 20 meters, but there were some cases where the distance between coordinates assigned to a station represented kilometers. To fix this inconsistency, we manually fixed the coordinates for more than 50 stations with the help of the interactive google maps that can be found in [3].

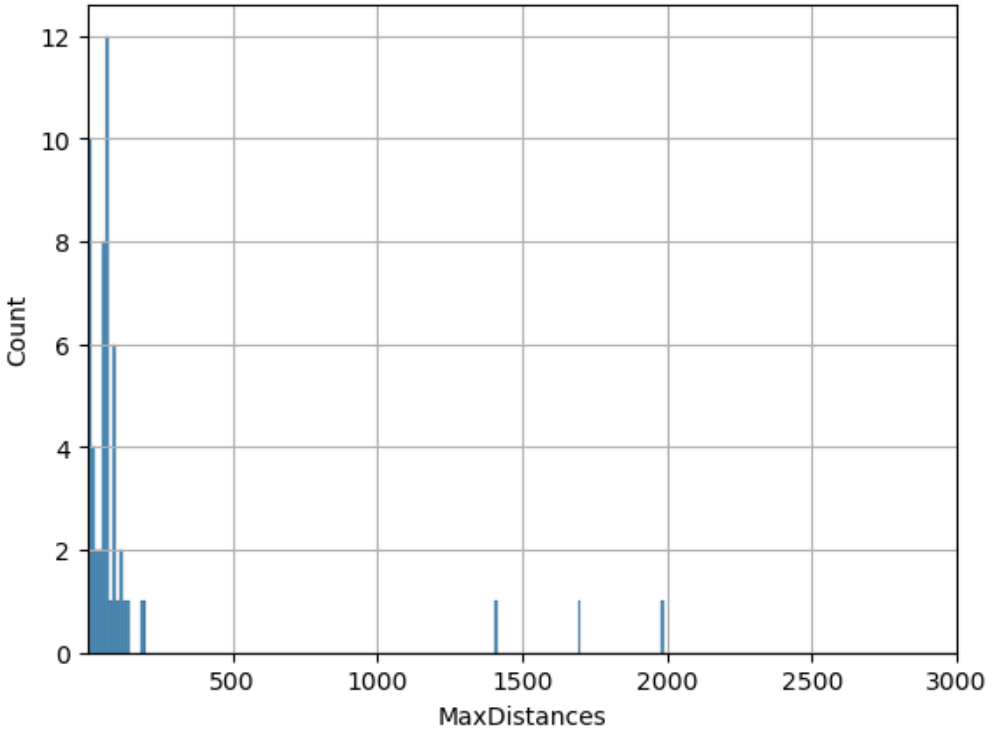
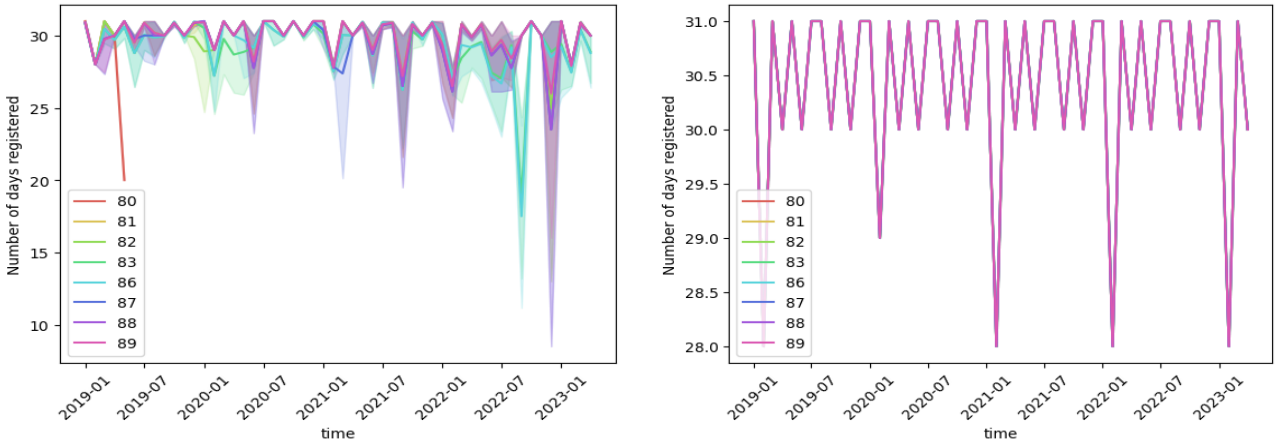


Figure 20: Counts of the stations per coordinate points max distance

6.5.3 Weather stations registers inconsistencies

When processing the weather data, the intention was to get the weather measurements per magnitude and per hour at each bike station. To do so, we used the coordinates to relate each bike station to the closest weather station, so that weather measures were as precise as possible for each bike station. The problem came when we tried to apply this approach to the complete time span. For some reason, stations registering was not consistent over each month. Each month, some stations did not have any results for some magnitudes, or they only had some days of the month registered. We suppose that this was due to maintenance on those stations or failures. This behavior can be observed in figure 21a which shows the dispersion of days registered in each month in the different weather stations per each magnitude, we can see how from some months some stations only registered 10 days of data. A better approach for this database, would be to remove all stations that do not contain the whole month registered, so that working with this data gets easier.



(a) Weather stations registering with inconsistencies

(b) Consistent weather stations registering

Figure 21: Weather stations inconsistencies

6.5.4 Final Dataset

After wrangling the various data files scraped from the internet and combining them, we started to assess the quality of the final combined dataset for station status at every day and hour and trip transactions. There were a small fraction of nulls that were easily imputed. We encountered trip durations of thousands of minutes and of negative values, so we deleted any trips with durations less than 0 and more than 5 hours. It was during this process that we realized that the trip durations in the json files was in seconds and the ones provided in the csv files was in minutes. So we had to remake our dataset accounting for this. One of the biggest time sinks was incorrect assignment of ids across the various data files. There were multiple id columns associated with the bikes stations, but we first picked the value based on the id that was being used in the critical plug station and unplug station columns of the trips dataset. As a check we decided to corroborate everything with the station name and location. However, the station names were only in the csv datasets. So edited our aggregation code and remade the dataframe with this information. Based on this analysis we realized that the ID we were going to pick did not correspond with the station's true ID. The station name was correct but the ID selected

was wrong. In some of the station names the correct ID was included in the text. Therefore we wrote code to extract this id from the text of the and assign it to its proper station. However, we also encountered that some stations had multiple names. The image below exemplifies the issue:

```
# the problem: ids do not match the source of truth based on station name
print(trips[trips['station_lock'] == 116]['lock_station_name'].value_counts())

# some ids are even tied to 2 different station names (and ids)
print(trips[trips['station_unlock'] == 5.0]['unlock_station_name'].value_counts())
print(trips[trips['station_lock'] == 5.0]['lock_station_name'].value_counts())
```

```
lock_station_name
110 - Moncloa    12544
Moncloa           7405
Name: count, dtype: int64
unlock_station_name
4 - Malasaña      12608
Malasaña           7805
Fuencarral          186
5 - Fuencarral       21
Name: count, dtype: int64
lock_station_name
4 - Malasaña    12529
Malasaña         7947
Name: count, dtype: int64
```

Figure 22: Depiction of the ID Problem

In stations where this was the case, we selected the mode ID from extracted from the station name as that station's ID.