



Escola de Camins

Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Demand prediction of public bike sharing services using machine learning algorithms

Treball realitzat per:

Jaume Torres Bonet

Dirigit per:

Francesc Soriguera Marti

Màster en:

Enginyeria Camins Canals i Ports

Barcelona, 14-05-2019

Departament de Transports

TREBALL FINAL DE MÀSTER

Abstract

Bike-sharing systems have become a popular mean of transportation in cities in the last several years, with an increased investment in the number of docking stations and bikes, as well as the introduction of new dockless systems. These systems come with the main problem caused by demand imbalance, which is optimal relocation of bikes into the different docking areas with a limited number of vehicles and time. The first step to optimize the relocation of bikes in the system is to be able to predict the number of bikes in a particular docking station and particular time, which can be done through the use of machine learning algorithms.

Nowadays, with high amount of data generated by the biking systems, it is possible to use AI to discern the demand patterns and be able to predict future demand, which could then be applied to solve the relocation optimization problem. Several studies have designed different type of algorithms to tackle this forecasting, but there is a lack of a unique methodology to follow, as well as studies that use the same algorithm in different location to test for adaptability.

Therefore, this study aims to design a common methodology for demand forecasting in bike-sharing systems. To do that, several study cases will be carried out, taking into account different cities, as well as different aggregation levels, including city-level analysis, cluster, and station. Three different algorithms are proposed to carry out this task: Random Forest, Gradient Boosting and Neural Networks, all been previously optimized for this objective. Finally, the obtained results are evaluated by means of pre-defined error functions, which makes it able to compare the different scenarios analyzed.

Key Words: bike sharing, demand prediction, machine learning, neural network, random forest, gradient boosting.

Index

Index	1
Chapter 1: Introduction	3
1.1 Project Overview	3
1.2 Objectives	4
1.3 Thesis Overview	4
Chapter 2: Literature Review.....	6
2.1 Bike Sharing.....	6
2.1.1 Historic Background.....	6
2.1.2 Externalities.....	7
2.1.3 Demand asymmetry and relocation	7
2.2 Machine Learning	8
2.2.1 Random Forest Regression.....	8
2.2.2 Gradient Boosting Method	10
2.2.3 Artificial Neural Networks	10
2.3 Demand Prediction.....	12
Chapter 3: Methodology	14
3.1 Datasets	14
3.1.1 Weather Data	14
3.1.2 Bike-sharing Data.....	17
3.1.3 Data division and feeding.....	20
3.2 Prediction Levels	22
3.2.1 City Level.....	22
3.2.2 Cluster Level	22
3.2.3 Station Level.....	22
3.3 Clustering.....	23
3.3.1 Effects of clustering on data processing	23
3.3.2 K-Means	23
3.3.3 Effects of clustering in prediction accuracy	27
3.3.4 Clustering conclusions.....	30
3.4 Tested Algorithms	30
3.4.1 Random Forest Regression.....	30
3.4.2 Gradient Boosting Method	32
3.4.3 Deep Neural Networks	34
3.4.4 Model comparisons	36
Chapter 4: Case Studies	38
4.1 New York	38

4.1.1 Data exploring	39
4.1.2 City – level prediction	43
4.1.3 Cluster – level prediction.....	49
4.1.4 Station – level prediction.....	52
4.2 London	63
4.2.1 Data exploring	63
4.2.2 City – level prediction	68
4.2.3 Cluster – level prediction.....	72
4.2.4 Station – level prediction.....	77
4.3 Overall Results.....	87
Chapter 5: Conclusions and Future Work	88
5.1 Conclusions.....	88
5.2 Future Work	89
References.....	90

Chapter 1: Introduction

1.1 Project Overview

Bike-sharing systems became popular in bigger cities some time ago, where there was a need of decreasing road congestion and also provide the citizens with more means of public transportation besides the traditional bus and subway systems that could solve the “last mile” problem. They are now mature systems with their own positive and negative externalities.

One of their difficulties is how to deal with the relocation problem. This consists in predicting where will bikes be more needed and less needed, to then move them from low-usage areas to high-usage ones. This problem increases in difficulty considering the imposed spatial constrictions, such as limited number of docks in the stations, limited number of bikes, and limited capacity concerning the vehicles used for relocations.

Besides coming up with better algorithms that minimize the cost of relocating the bikes, another way to improve the management of bike-sharing systems is to be able to predict, with enough time, where and how much demand will there be in the system. To do that, simple algorithms based on recent demand patterns were used to extrapolate future demand, with final decisions not being as accurate and realistic as they should be.

All this changed with the introduction of *Big Data* and *Machine Learning* algorithms such as Neural Networks. Nowadays, it is easy for bike-sharing systems to generate a high amount of data, which have become susceptible for analysis thanks to the all-time-high usage of algorithms based on AI. This fact, along with the highly patterned demand use made it possible to predict usage patterns more accurately and realistically.

The problem with these algorithms is that they are extremely accurate in localized systems, which implies that completely different algorithms could be used in cities that are not that different in terms of demand patterns. On one hand, this absence makes it possible for a lot of different systems to be developed, which then can be compared against each other to improve other systems. But, on the other hand, this means that new studies do not have a general idea on what steps should they follow to develop a new predicting model, since there are so many different approaches. Therefore, there seems to be a lack of generalization or common methodology on what variables to take into account or what are the procedures to follow to reach minimum accuracy levels.

1.2 Objectives

The main objective of this master thesis is to consolidate a proper methodology to forecast bicycle demands in different aggregation levels: city, cluster and station.

There have been several studies that try to predict the trips generated and received by bike-sharing stations, but each one of them specializes in some city in particular. This means the accuracy of the predictions reach a reasonable level, since specific spatial and time distributions in the metropolitan are tackled in building the algorithm. On the other hand, there is not much consensus on how to proper deal with demand forecasting in different bike-sharing systems, and whether a same methodology can be used to tackle different case-studies.

Therefore, this master thesis tries to bridge these gaps in the different methodologies, not only applying the it to different bike-sharing systems but to different aggregation levels as well. The same error metrics have been used among the carried case-studies to be able to fully understand the problematics each city has and be able to properly compare the results.

1.3 Thesis Overview

This master thesis will be divided in eight chapters, the first one being the introduction and the last one being the overall conclusions and future work, with an extra chapter for references.

Chapter 2 will introduce an extensive literature review, first explaining how bike-sharing systems came to be, their objectives and characteristics and their current situation, providing examples of different systems around the world. Once introduced, their advantages and disadvantages will be pointed out, and how demand prediction is able to smooth out some of their problems. Next, an introduction on machine learning, the methodology used to forecast the demand, will be reviewed. First, the main idea behind these methods will be introduced to familiarize the readers with the inner works of these kind of algorithms. Then, several types of machine learning algorithms will be introduced and explained in more detail, such as Random Forest (RF), Gradient Boosting (GBM) and Artificial Neural Networks (ANN). Finally, the introduction to bike-sharing systems and machine learning will be tied up, analyzing the application of these methods in the bike systems and the advancement this has supposed.

In Chapter 3 the datasets used to describe the demand behavior of bike-sharing systems are presented, with the basic hypothesis taken into account. An extensive analysis on the data quality and linearity is carried out, providing ways to overcome problems when scrapping data. A clustering methodology to treat the stations is introduced, with an explanation of the K-means

method and the variables used to group stations together. Afterwards, different prediction levels of this study are introduced in more detail, focusing on the city-level predictions, the cluster-level and the station level. The advantages and disadvantages, as well as their utility, is presented for each case. Then, each one of the algorithms used to forecast the demand are introduced with greater detail. First a small introduction on the particular algorithm is given, as well as its advantages and disadvantages. Then, the parameters to optimize are introduced and different tests are carried out to see which configuration is optimal, considering accuracy and computational time. This process is repeated for a Random Forest model, a Gradient Boosting one, and a Deep Neural Network. At the end, the overall performance of the models is compared, with its pros and cons.

Chapter 4 introduces the results obtained when the optimized algorithms are applied, presenting two different case studies corresponding to two different cities: New York and London. Their demand patterns are analyzed, and a brief introduction on each of the cities' weather particularities is presented. Then, different levels of prediction are carried out for each of the cases studies, comparing the performance of the algorithms and the accuracy of the predictions.

On Chapter 5 the overall conclusions are presented, reflecting on the different case studies analyzed and on the overall prediction methodology. Furthermore, possible future work using the presented forecast methodology is mentioned.

Chapter 2: Literature Review

2.1 Bike Sharing

In recent years, there has been a shift towards “sharing economy”, which is challenging traditional ideas of ownership and consumption. In particular, transport sharing systems have become increasingly popular around the globe. This kind of systems try to remove the need of ownership over a private vehicle (car, bicycle) to instead sell the access to them, somewhat similar to public transportation [1].

2.1.1 Historic Background

Even though their popularity has increased in the recent year with the introduction of new innovations, what is considered the first bike-sharing system was established in Amsterdam in 1965, with what was called “White Bicycle Plan” [2], where bikes were painted white and left unlocked for everyone to use, supposedly to be used for a single trip [3]. They heavily suffered from theft problems and were therefore not viable as a reliable transportation system.

In 1995, the first urban level bike-sharing program (*Bycyklen*) was introduced in Copenhagen, where users paid a refundable deposit and had unlimited use of the 800 bikes in a certain zone [4]. The system tackled theft problem by imposing fines to users that did not return or damaged the bike, which was found to be a successful strategy.

Trying to improve the coin-deposit stations, *Bikeabout* was launched in the same year. It consisted of automated stations and magnetic cards given to students of the campus site that was used to unlock the bike docks. The system did not find success, mainly because of weather restrictions and limited number of bikes [5]. However, similar systems were proved to be useful in many other cities, where they are still used nowadays with some innovations such as GPS systems, e-bikes or smartphone applications.

Finally, dockless bike-sharing systems can be defined as the fourth generation of bike-sharing, where the docks or stations bikes were usually placed were replaced for self-docking bikes. These systems started to be used in large scale in Asia, where there is a tradition of bike usage in cities, but there were lacking a docking bike-sharing scheme [6].

2.1.2 Externalities

There have been several studies focusing on the possible externalities of transport sharing systems, for both car sharing systems [7] and bike-sharing ones. The last ones in particular have been a quite popular topic in transportation journals, with the number of articles related to them exponentially increasing since 2010 [8].

The main objective of bike-sharing programs is to promote sustainable transportations, mainly in cities, and tackle the “last mile” problem, therefore providing an eco-friendly alternative for commuters hoping to reduce private vehicle use and traffic congestion [9]. On the other hand, new systems (especially massive dockless bike schemes) have been lately criticized for waste of resources and uncontrolled parking [10].

2.1.3 Demand asymmetry and relocation

One of the main difficulties to assess is the resource relocation problem, which affects both docked and dockless bike systems. This consists in the lack of demand equilibrium in the city, with areas heavily generating trips, whereas some others receive it. These different areas have been successfully identified in past papers, with office areas being responsible for attracting more demand during the morning peak and generating more trips on the afternoon peak, whereas residential zones act the inverse way [11]. Other identified zones are intermodal connection zones or areas with high density of bike lanes [12].

This demand asymmetry makes it quite impossible to think of these schemes as self-sustainable, and extra effort has to be put to relocate the bikes depending on the demand if a more efficient system is wanted [13]. Several studies have studied the user’s behavior to try to understand the mechanism behind the system unbalance [14] [15], with two main approaches: User rebalancing and operator rebalancing.

For a bike-sharing scheme to be self-sustainable, the users themselves must be able to rebalance the demand in the system with none or little help from the operator. This has been studied in several papers, where different incentivizing approaches were applied, such as price discount or other type of advantages [16] [17] [18].

On the other hand, there has been a surge in studying the relocation problem from the operator perspective, which can be simplified to a mix between a Knapsack Problem [19] and a Travelling Salesman Problem [20], which both settle the base for most of the logistic problems involving inventory limitation and route optimization. Since both are quite difficult to solve in large scales (the first one is a NP-Complete problem, whereas the second one a NP-Hard, several powerful algorithms have been especially developed to solve them. In the particular problem that

is to optimize the allocation of bikes in the different stations of a bike-sharing systems, different research have had success in different degrees and different focus, such as route optimization [21], inventory optimization [22] [23] or both [24]. The main characteristic shared between these different research papers is that all of them used the great amount of data provided by bike systems to either base their hypothesis on them or directly train the algorithm against them using machine learning methods [25].

2.2 Machine Learning

Machine learning is just a broad term to refer to a series of algorithms that are characterized for not having clear instructions and instead relying on mathematical models based on input data. This data is usually separated in three parts, consisting on:

- Training set: The dataset is used to train the model and adds up to most part of the data (between 50 to 80%). Both input and output are present in the training dataset, since they are necessary to calculate the error and “learn” from the mistakes.
- Validation set: Used to assess the error in the model once it is trained. Input and output are also provided.
- Test set: Dataset used for the actual predictions, where the model is already trained. It is usually considered a sub-set of the validation set and the last one to be tested.

A high number of different algorithms have been developed under this term, more in particular under the umbrella of supervised-training algorithms, but this literature review will focus on the three main ones used for the research: Random Forest, Gradient Boosting and Neural Networks.

2.2.1 Random Forest Regression

Random Forest (RF) is a type of algorithm that can perform regression and classification tasks as well first created in 1995 [26]. They fall into the ensemble learning algorithms, which consist on averaging multiple learning algorithms with the objective to obtain higher accuracy in the final prediction. In particular it is considered a bagging method, where the predictors are independent between them and are then combined using some averaging technique [27].

Thus, the basic idea behind a RF is to create a tree of decisions based on a sample data. Each of these decision trees can be understood as a single algorithm by themselves, consisting of their own nodes and branches (Fig. 2.1).

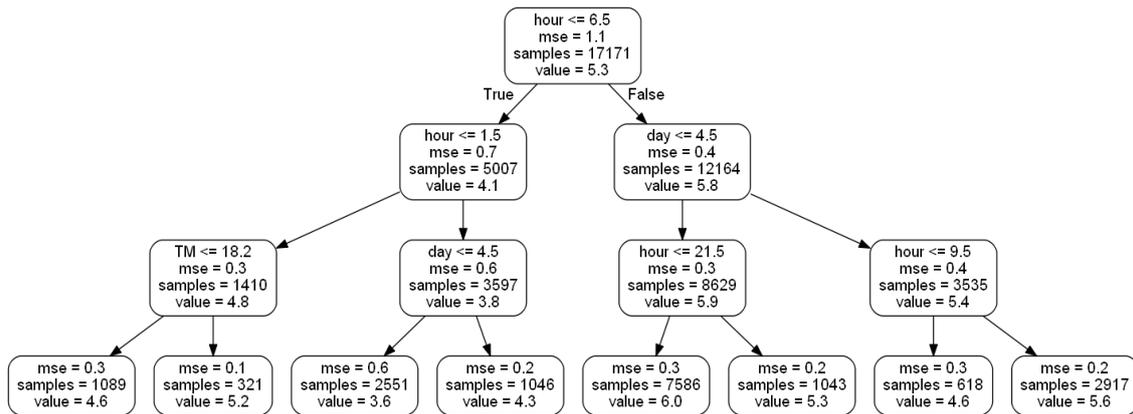


Fig. 2.1 Example of a decision tree included in the used Random Forest algorithm.

On every node of a single tree, the input values are checked against input features. The decision algorithm on which feature to split first depends on the user, but usually the RF chooses the best feature based on minimizing the error on each node. This decision algorithm would make the different trees quite repetitive, but not all the features are considered in each tree, which adds the extra randomness needed in the method.

When the input value is checked in a single node, the output is either ‘True’ or ‘False’. From there, the input values are sent to the next node and so on. This is done until the leaf-node is reached, that is the last node, where no more splits take place. The maxim-depth of the model can be used as an input, but error functions are usually left to do that calculation automatically for each tree, since different features are used.

Once every tree is calculated and the forest is completed, each of the estimators ‘vote’ and the most voted result is chosen. This is done for each input value and the decisions used to construct the forest (features to use, number of nodes to split...) are saved to carry out future predictions. At the end, we end up with a robust method capable of classifying and predicting variables given that the features included are enough to describe their variance.

2.2.1.2 Advantages and disadvantages

The main advantage of a RF is how easy it is to implement. With only characterizing the number of estimators and the maximum number of features one can easily get a somewhat accurate model. It is also quite versatile regarding inputs, since it easily accepts categorical variables as well as continuous ones, not requiring standardization in most of the cases. Since the randomness is kept in every tree and then the solution is done by averaging, it is difficult for this kind of algorithm to overfit.

The main disadvantage is its ‘black box’ effect. In juxtaposition to not requiring a lot of effort from the user perspective to get some results, it is also difficult to improve them tweaking the

hyperparameters, since you are never sure how is the forest really progressing.

2.2.2 Gradient Boosting Method

Gradient Boosting is another type of ensemble method, but in this case belonging to another class: Boosting. The main difference between boosting and bagging methods is that the first one creates the predictors sequentially, instead of independently as bagging methods do [28]. Thus, predictors learn from the mistakes of the last iteration.

The observations are not chosen based on a bootstrap process, like a bagging method would do, but the ones that have higher error tend to appear the most. This means that these algorithms are more prone to overfit than bagging methods, where randomness is introduced in the predictors through independent sampling.

Like any other supervised learning algorithm, it minimizes a pre-defined loss function, in this case using a gradient descent method, to then update the estimators (trees) based on a learning rate parameter.

2.2.2.1 Advantages and disadvantages

One of the main advantages is that it reduced the bias and variance of the data, since being a sequential model makes it possible for the algorithm to get better in each iteration. This same fact makes it possible for this kind of algorithms to generally perform better than bagging methods.

This is also one of their main disadvantages, since there needs to be a clear stopping criterion, otherwise the model will overfit. As other ensemble methods, it also suffers from a ‘black box’ effect, where it is not clear how the algorithm is working.

2.2.3 Artificial Neural Networks

Artificial Neural Networks (ANN) are a machine learning algorithm inspired by the neuronal system of a brain [29]. It consists of a series of modules, in this case called neurons, that receive an input, which then is computed by a series of internal functions and is passed as an output to the next neuron. The structure of the neural network (NN) is determined by the number of neurons and number of layers, as well as the objective error loss function to minimize and the activation functions in each layer.

At first, the input values are fed into the neural network on the first layer, previously standardize (in case of continuous variables) or hot-encoded (for discrete variables). This input is then multiplied by a weight value and feed forward to the first hidden layer, where the transformed input goes through the activation function of each neuron.

This activation function is defined for each layer, rather than for each neuron, and it consists

of a simple function that transforms the input into an output value. There are basically two types, with sigmoid functions (*tanh*, *softmax*, *softplus*...) or linear ones (*ReLU*, *linear*...). Once the output is calculated, this value is again feed forward to the next layer multiplied by a different weight value. This process is done until the output layer is reached (Fig. 2.2).

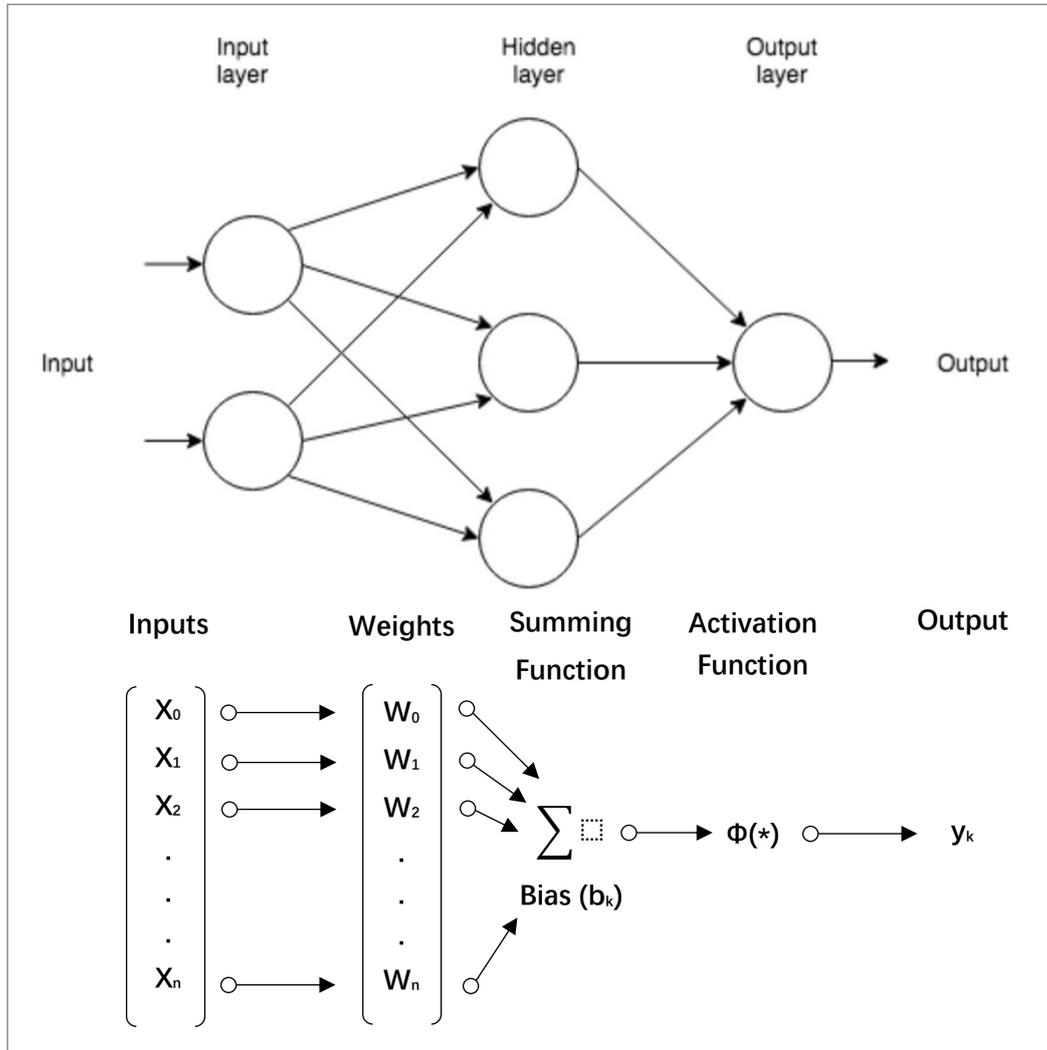


Fig. 2.2 Example of a Neural Network structure.

Once the output is reached, the error between the predicted value and the real one is calculated following the defined error function, which depends whether the NN aim is to act as a regressor or classifier. For the first time, RMSE is usually used, whereas for the second one maximum-likelihood or cross-entropy is preferred.

The key in the learning method of the NN is in the next step, the backpropagation of the error onto the weights [30]. This is done by what it is called an *optimizer*, which is an algorithm that adjusts the weights. Well-known optimizers are *ADAM* and *SDG*.

2.2.3.2 Advantages and disadvantages

Unlike Random Forest or Gradient Boosting, Neural Network optimization can be much more complex and complicated. This is due to the fact that not only you have to tune some parameters, but also build the structure of the algorithm itself. This is, in itself, an advantage and an inconvenience.

On one hand, part of the “black box” effect of ensemble methods is lost, since you also are designing the layout of the algorithm. On the other hand, this increases the complexity of the model, since it is not as easy to just input your data and get a decent output.

This complexity comes with another advantage, which is that since you have more control on the structure of the algorithm and its parameters, it can further be improved to get a better accuracy compared to other methods.

Finally, another disadvantage is the easiness in overfitting, which depends on the number of epochs (the number of times the algorithm sees the same data). The more epochs, the more the algorithm tends to memorize the outputs, rather than learn the relations between them. This means that during the training set, the error will decrease with each step, whereas on the test set, since it will not have any information of that data, the error will actually increase. To tackle this problem, one should be careful to control the number of epochs and, if necessary, add dropout layers.

2.3 Demand Prediction

Several papers have used machine learning algorithms to try to predict the demand in different aggregation levels. From the reviewed papers, different variables can be considered to affect the demand on bike-sharing systems, besides hour of the day, day of the week or other time-related variables [31].

Most of the papers agree that weather is heavily correlated to the demand on this type of systems [32]. In particular, rain or other forms of precipitation can heavily lower the demand during that certain time step, whereas temperature has a smoother effect, with colder seasons generally exhibiting a lower demand, whereas temperate seasons showcase higher demand [33]. Other variables that are considered in a broader region are the number of slopes between stations and some zoning information, such as percentage of residential area or green spaces [34].

Once the data is correctly treated, some research papers group similar stations together based on their similarities, using a nearest K-neighbors approach [35], Voronoi regions [36] or community structures [37].

After that, several algorithms per paper are compared against each other and conclusions are reached. In general, there is a worst prediction accuracy for stations or regions with less demand, since data is sparser [38]. There is not a single standout algorithm which works best in most of the cases, since different stations in the same system might have completely different behavior, but it is concluded that machine learning algorithms are capable of explaining most of the variance in bike-sharing demand [39].

Most of the papers reviewed directly output the predicted value, but this makes it impossible to determine any kind of confidence interval. Even though this is usually done through Bayesian approaches [40], this is also possible using only Neural Networks by means of dropout layers. Dropout layers were first introduced in 2014 [41] to prevent overfitting, which basically consists on dropping a neuron during training set to reduce a network generalization error. It has been shown that adding a dropout layer before every dense layer in a NN, in both training and testing set, can be interpreted as a Bayesian approximation in deep Gaussian processes [42], where the multiple passes to obtain the different output values can be understood as a Monte-Carlo Sampling.

Chapter 3: Methodology

3.1 Datasets

In any type of research that involves Machine Learning the base is always the quality and quantity of data you can collect for the project. Bike-sharing systems are characterized for being susceptible to weather conditions, since they are usually outdoors. Thus, besides the actual data from the bike systems, climate data on the cities that are analyzed also has to be collected.

Each city analyzed has at least one bike-sharing system that provides open-data for users to analyze. This information might come from the company in charge of system, such as Citi Bikes in case of New York City or from the city hall itself, such as Bicing in Barcelona or Santander Bikes in London.

As for the weather data, it is recommended to extract it from the same source, rather than separately obtain it for each of the cities analyzed. It is important to choose not a trustful source but also convenient one. In this case, the *DarkSky* API has been used to collect the needed data, since it can be easily integrated into *Python*, provides an hourly historic record and, most importantly, has the same structure for every different location. This last feature might be the most important one, since it makes it possible to generalize part of the workflow process.

Besides weather and bike-sharing data, time data is also used, as it ties together both datasets. Season, day of the week, hour and minute have been extracted from the datasets to be included in the analysis. Two more variables, in this case binary, are also taken into account to take holidays into consideration. One considers the national holidays, whereas the other one also considers the students holiday. It was decided to consider both to be able to explain the fact there is less demand during certain long periods of time, such as summer or Christmas, where schools are off but people might still go to work.

3.1.1 Weather Data

Since the data regarding the climate is taken from the same source, the structure remains the same for each city analyzed. The data used for this project ranges from January 2017 to January 2018, therefore a whole year of data needs to be processed.

3.1.1.1 Analyzed Variables

There are many climate variables one can analyze, but the ones most prone to affect user's mobility are chosen. This pre-selection is just based on one's judgement and for sake of simplifying the problem, so there might be more relationships between weather – mobility that are left to analyze. In this case, the variables taken into account are:

- Temperature: Average temperature in the chosen time-step (1h) in Celsius.
- Wind speed: Average wind speed in the chosen time-step (1h) in m/s.
- Precipitation: Distinguishing two types between snow (cm) and rain (mm).

Once all the needed data is retrieved, it is always necessary to check for errors, null values or not expected distributions. This can be simply done by plotting the data against time as well as their histograms and density functions (Fig. 3.1, Fig. 3.2).

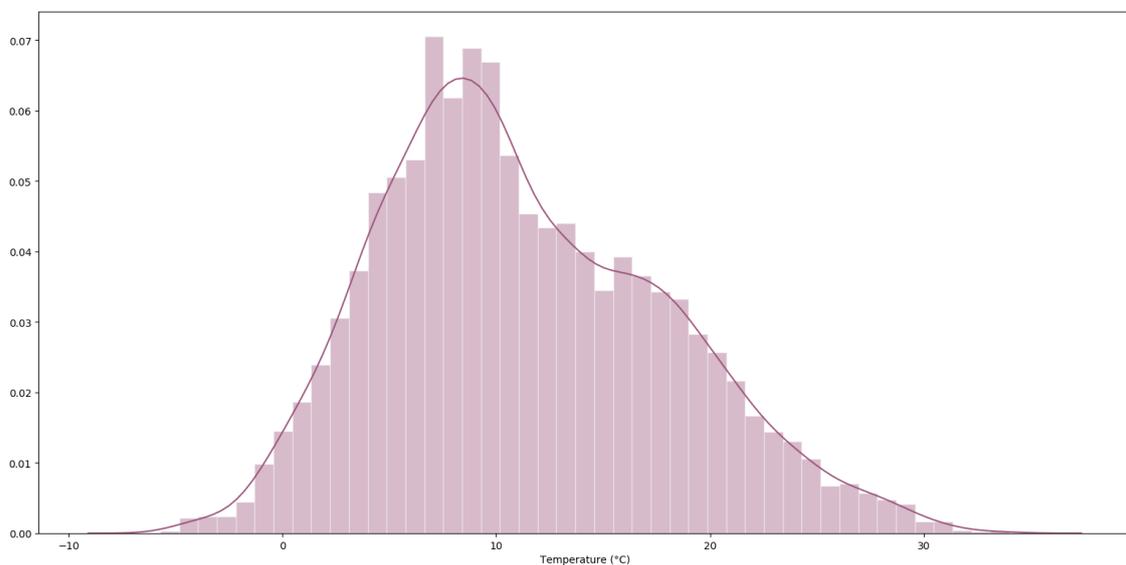


Fig. 3.1 Temperature Histogram for London City.

For instance, in case of London's temperature, we see that the distribution is as expected (approximately following a normal distribution). As for the values themselves, we see four suspicious values between July and September 2018. These take the value of 0.0 °C, when the temperature right before or after that exact time step is around 25 °C. In these cases, the mean value between the consecutive time-steps is taken to replace the erroneous values. In the same way, similar analysis can be carried out for the other weather variables.

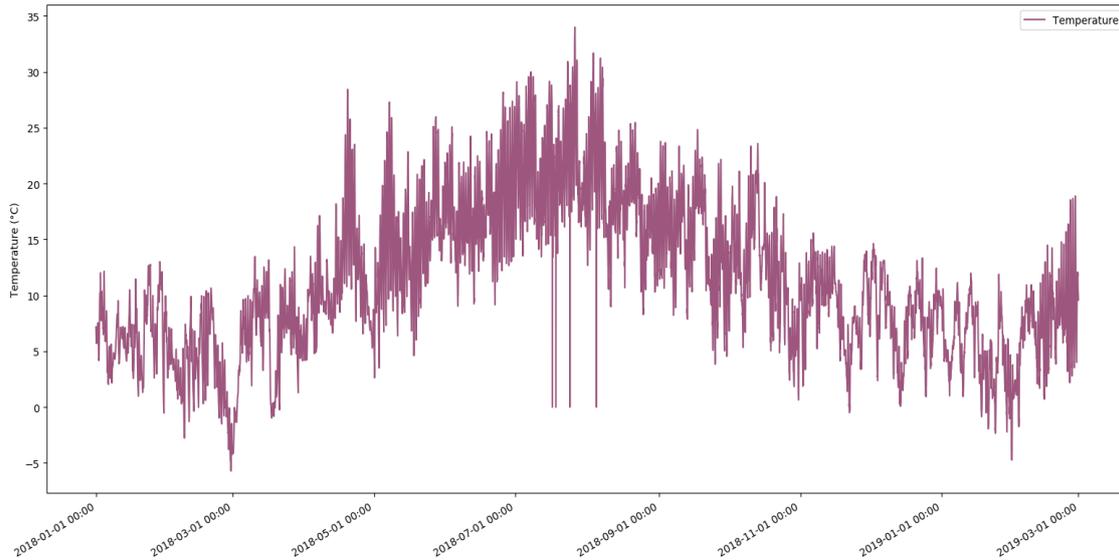


Fig. 3.2 Temperature evolution during a year in London.

As for the wind-speed variable, there are quite a lot of times where a 0.0 speed is registered, which is not that realistic given that both the previous and next time-step that value is quite different from zero. This might be mainly due to measure errors, but instead of applying a mean value it is chosen to leave the zero value, since low wind speeds do not affect the model in any way.

Finally, rain and snow are not that common, thus when they are present they will have a great effect in the demand modelling. For example, from the whole year of data available corresponding to New York only 16,32% of the registers have some kind of precipitation, representing the 50,41% of the days registered and with an average value of 0,725mm for rain and 0,156cm of snow.

3.1.1.2 Discretization of the variables

Once all the weather data has been filtered, one must think whether any transformations are necessary so our model can correctly read the data. For numerical data, the most direct approach is to standardize the variables and feed them into the model. This usually works fine, but since the weather data used for predictions has some kind of error, the value itself might be incorrect and confuse the model. Therefore, a discrete approach has been chosen to deal with weather variables.

Discretization involves classifying the values onto certain ranges, therefore having several classes for such variables. The range can directly be determined from a maximum – minimum approach (dividing the range into the number of classes one needs) or from a probability function. This second approach is the chosen one, since the model is capable to adapt better to classes that

have the same number of registers.

This is directly applied to variables like temperature and wind speed, distinguishing five different classes. As for rain and snow, a single class has been chosen to represent the zero value (lack of precipitation), whereas for other values different than zero the discretization is applied as usual. Since these are now categorical variables, rather than continuous ones, each of their values is assigned into a different variable, one variable for each category (hot-encoding).

3.1.2 Bike-sharing Data

Each city's system is organized different, with more or less variables included and different time precisions, but the provided datasets can be classified into three types:

1. Station – based: Includes historic information on the stations included in the bike system, such as available bikes, available docks, capacity or availability. Each row is a different station at a certain point in time, with the time frequency being constant.
2. User – based: Includes historic registers of the user movements, with information such as trip duration, start station, end station or chosen bike. Each row is a user trip at a certain point in time, with non-constant frequency.
3. Live data: This includes the station – based information but at the exact time the data is collected and it is updated every minute.

3.1.2.1 Analyzed Variables

The main feature to analyze from the bike-sharing dataset is the number of trips generated or received from a single station, therefore the user-based data will be the base from most calculations.

Having said that, it is also useful to have station-based stations to detect outliers, which is the first step to cleaning the bike-sharing dataset. These are stations that are already in the system but are not open for users, therefore they barely have received or generated trips. To do that, we select all the unique stations that appear in the user-based table and match them with the station-based data we have. From this join, we obtain which stations have no data or the number of using them are so low they can be considered outliers. In the case of New York, this can be easily checked using *Tableau* to visualize which stations we do not have data, which turn out to be the stations based in New Jersey (Fig. 3.3).

Once the outlier stations have been taken care of, one must look into the user-based register and look for suspicious data. In this case, rather than visualize we will apply basic *SQL* to get rid of this outlier data. We will be looking for extremely long trips and trips with a duration of less than a minute. Once these trips are deleted, we can consider the dataset to be clean.

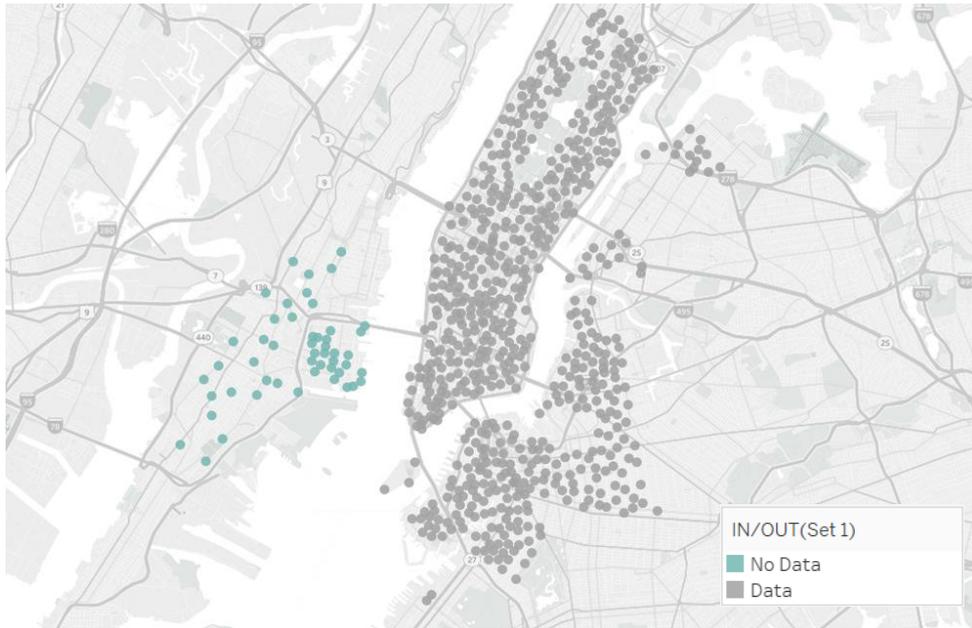


Fig. 3.3 Outlier stations (No data) in New York City.

Since the information of each trip is given at the time of start, we need to aggregate the data in a certain way so the time-step is constant. This time-step is chosen depending on the analysis level one wants to carry out. For a city-level prediction, 15min provides enough information and accuracy. On the other hand, in the station-level prediction, the time-steps will range from 1 hour to 2 hours, since there is much more variability in the data which needs to be aggregated to become as regular as possible.

It is recommendable to further explore the data before applying any kind of algorithms. This involves plotting the number of bicycle users over time and analyzing the average number of users per day of the week.

From Fig. 3.4 we can see two clear distributions of user trips during the week, in this case exemplified in the case of Barcelona. One happens during weekdays, with a clear peak around 8am and another one at 18pm, corresponding to work shifts. On the other hand, we see a much softer curve during the weekends. Friday is an exception, having a weekday distribution during the first hours, to then have a distribution similar to the rest of the weekend. These features can give us ideas on how many time variables to take, in this case we might consider taking Weekday – Friday – Weekend variables instead of one for each day of the week.

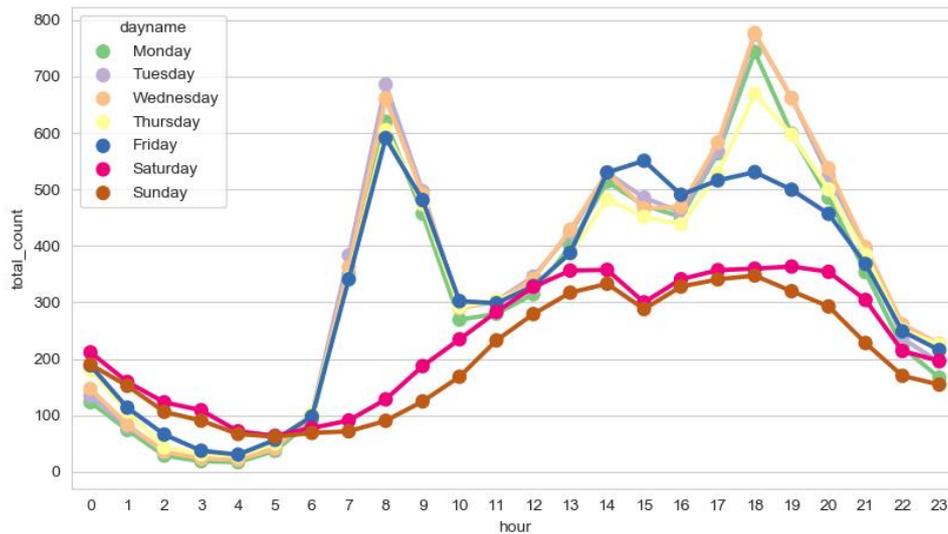


Fig. 3.4 Bike usage distribution depending on the day of the week in Barcelona.

We can make the same analysis buy seasonally-wise (Fig. 3.5). We can observe that, in the case of Barcelona, we only have data from three seasons (summer, fall and winter), but they have quite different patterns. Surprisingly, summer and winter have almost the same usage, whereas fall has an overall bike usage. This might be due to the fact that the *Bicing* system is not generally open for tourist use (only subscriptions are allowed to use the bikes), thus only locals will be using them. Thus, if people generally used them for commuting to work, there will be less of a use during summer and winter, where most vacations are situated and the weather might be too extreme. We can expect for spring to act similar to fall if this pattern is true.

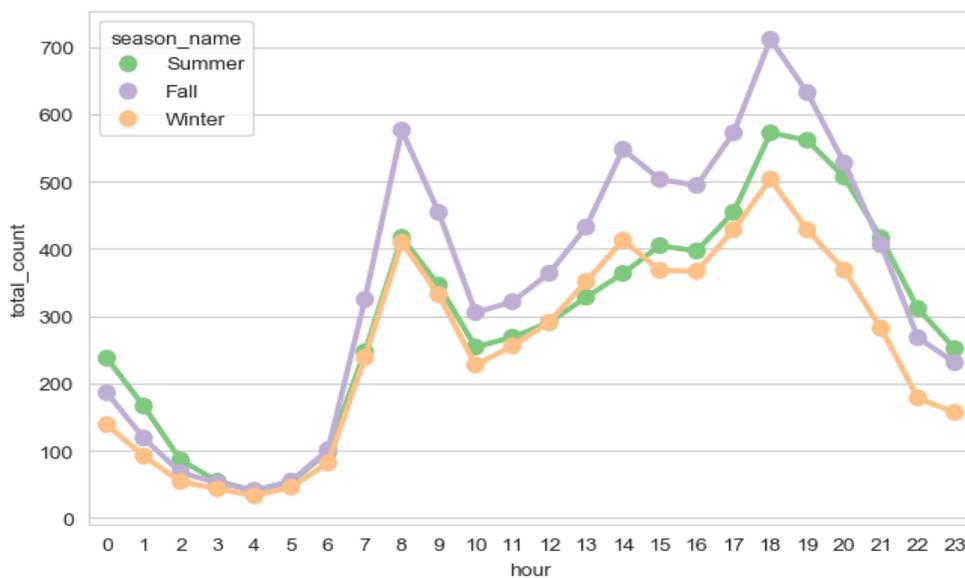


Fig. 3.5 Season hourly distribution of bikes in usage in Barcelona.

3.1.2.2 Data processing

As mentioned before, the bike-sharing dataset includes two variables, being the number of trips generated and the ones received received in a certain station during a certain time-step. Since this variable can be considered to be continuous and it is supposed to be the output of our model, we standardize their values before feeding them into the model. That is, subtracting the mean and

$$X_{st} = \frac{X - \mu}{\sigma} \quad (3.1)$$

dividing by the variation (Eq. 3.1).

3.1.3 Data division and feeding

Once the weather data and the bike-sharing data area cleaned, one can join them onto each other using time as the index, given that they have been resampled using the same time-step. Doing that, all the data used for the simulation can be resumed onto a single table. As a last step, it is also recommended to calculate the covariance matrix to remove any linear-dependent variables that might still appear in our datasets.

Once the data is completely treated, resampled and the categorical variables have been hot-encoded, the dataset has to be divided into two, one being the training set and the other one the testing set. The first one represents a random sample of the whole set corresponding to the 80% of the whole amount of data, whereas the testing set represents the other 20%. The training set is used to train the algorithm by calculating an output, then comparing with the real result and backpropagating the obtained error to keep training. The test set is used in parallel, to check whether the algorithm is learning the hidden relationships properly by just predicting this set of data and calculating the error.

Besides the training and testing set, a validation set might be used as well. This corresponds to data the algorithm has never seen before, therefore corresponding to an actual prediction, even though the output is also known and used to calculate the error. In this study, the data from 2018 was used for the training and testing set, whereas part of the data available from 2019 (January) was used as validation set.

To conclude the explanation on the dataset and the variables used, the overall process for the data treatment is schematized in Fig. 3.6. It should be noted that for the bike-sharing data, not all the stations are used in the same model at once, but they are pre-selected based on cluster analysis, which is introduced in part 3.3. This yields to different prediction levels available to carry out, them being a city level analysis, cluster level or station level.

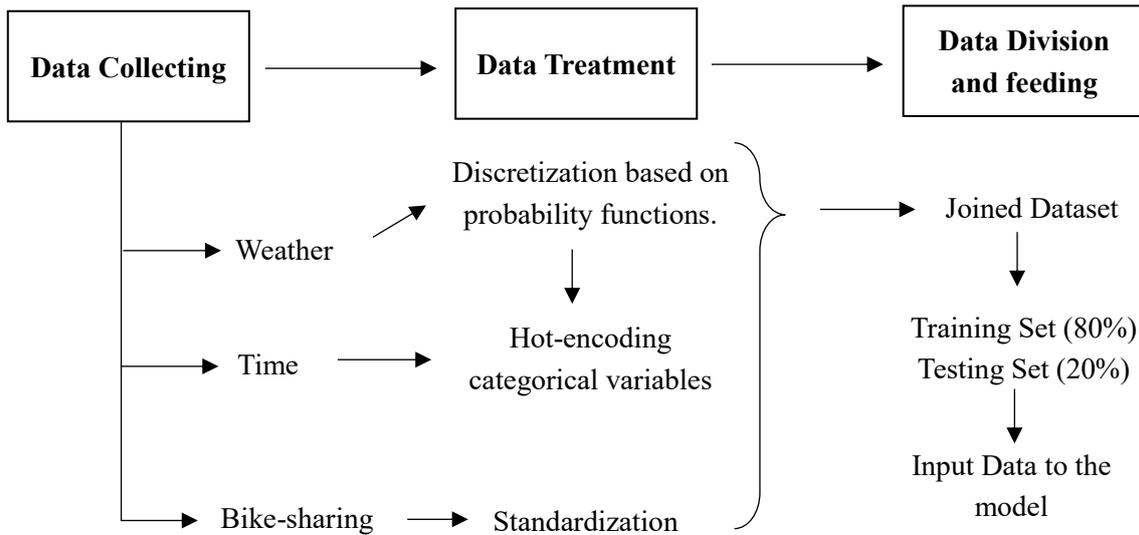


Fig. 3.6 Data collecting, treatment, dividing and feeding scheme.

Table 3.1 Variables used to predict the bike demand in the stations

Variable	Type	Range
Trips received	Continuous	Positive
Trips generated	Continuous	Positive
Station ID	Discrete	0 – Max(Stations)
Season	Discrete	0 - 3
Day	Discrete	0 - 6
Hour	Discrete	0 - 23
Minute	Continuous	0 - 59
Temperature	Discrete	0 - 5
Wind speed	Discrete	0 - 5
Rain	Discrete	0 - 5
Snow	Discrete	0 - 5
Holiday (National)	Discrete	0 - 1
Holiday (School)	Discrete	0 - 1

3.2 Prediction Levels

3.2.1 City Level

Once the data is retrieved, processed, and the algorithms have been optimized, a city level prediction is the first one to take into consideration. It consists on predicting the total demand of bikes in the city, both returning and renting.

The advantage of this kind of prediction is that you are working with data agglomerates, which tend to reduce the variability that cluster or station-level predictions might have. Since it is the sum of every station in the city, the amount of data is heavily reduced (a single data row for every time-step considered). This makes it easy and fast to check how the algorithms are behaving and grasp a general understanding of the demand patterns on each city.

3.2.2 Cluster Level

From the raw clean information, the next aggregation possible is on cluster level. This option has the advantage of having a more detailed view than a city – level prediction and a higher accuracy than a station-level one, since you are still predicting on aggregates.

Cluster information can provide general information on the number of bicycles demand in districts or particular zones, where it is not necessary to know exactly the behavior of the stations, just the overall number of bikes a group of close-by stations might need.

3.2.3 Station Level

This is the most detailed level of prediction possible in the study, as well as the less accurate one. One trying to predict the behavior of a single stations, much more variability comes into play, which increases the overall error.

To tackle this problem, a larger time-step is considered, with at least one hour for high-demand stations and two hours for low-demand ones, compared to the less than an hour time-steps considered in the other cases. This is still extremely detailed information that can be proven to be useful not only for the company in charge of relocating the bicycles but also for the users.

3.3 Clustering

3.3.1 Effects of clustering on data processing

One could think of after all the data is processed the process is as simple as feeding the data into the model and start the calculations, but more preprocessing is still needed. There are two main reasons for applying clustering methods in our data:

1. Too much data: If we were to train a single model with all the data at once, most probably the computer would not be able to finish all the calculations and fail.
2. Too disperse: If training into a station-level, data is still too dispersed to be training all the stations at once, since the algorithm would try to find relationships between stations that do not have much in common, therefore increasing the error and the convergence time.

If we only considered the first reason, it could be possible to carry the calculations in different computers in parallel, but the second reason should still be taken into account, since the data should be divided into different computers in a organized way, therefore taking data dispersion into account.

Hence, for these reasons a clustering technique is applied to the cleaned data to try to minimize the calculation time and error. In particular, K-Means has been chosen for the simplicity of the process and fast calculation.

3.3.2 K-Means

A K-Means method simply tries to minimize the distance between points to then group them together based on their similarity (shortest distance). There are different methods to define the distance formula that should be considered, but usually the Euclidean distance is used (Eq. 3.2).

$$d = \sqrt{w_a a^2 + w_b b^2 + \dots + w_m m^2} \quad (3.2)$$

where d is the compute distance, $a, b \dots m$ are the variables taken into account and $w_a, w_b \dots w_m$ are its respective weights.

The process the algorithm follows to define which points belong to each of the clusters is:

1. Input the points (x_1, x_2, \dots, x_n) and the number of clusters K .
2. Place the centroids C_1, C_2, \dots, C_K on random locations.
3. Calculate the distance between the points and such centroids.

4. Assign the points to each cluster by determining the minimum distance between them and its centroid.
5. Repeat the process from step 2 until none of the clustering assignments change.

3.3.2.2 Variables considered

We can consider as many variables as needed, but one should carefully think about what they want to achieve with clustering. In this case we are not only looking for stations that are spatially close to each other but also they have similar demand patterns. Since different variables have different magnitudes, it is important to scale them before doing any calculation. In this case, all the variables are scaled between 0 and 1.

To consider adjacent stations we must consider the station's location, in this case in form of latitude and longitude coordinates. We should transform them into UTM to be able to calculate the real distance between them, but since we are only looking for similarities, one can just scale them.

As for demand pattern similarities, there is a lot of data we can consider, but we will focus on the user-based information we have. This is basically due to the fact that reliable historic station-based information is not always available, whereas user-based is necessary to do any kind of demand prediction, therefore always available in our analysis. Hence, we considered the average generated and received trips during weekdays and weekends, dividing them into the first 12 hours of the day and the last 12. We consider the absolute values rather than a ratio, dividing for the capacity for example, to take into account the size of the stations as well, since larger stations will generate a higher amount of trips compared to smaller stations (Eq. 3.2). These values have been multiplied for their respective weights considering how many days of information they provide, this multiplying the weekday data by $5/7$ and the weekend one by $2/7$.

As for the weights, we apply a weight of 2 to the spatial data to make sure the stations will be as adjacent as possible, leaving a weight of 1 to the other variables.

$$x = (long, lat, G, R) \tag{3.3}$$

where *long* is the scaled longitude, *lat* the scaled latitude, *G* corresponds to the generated trips and *R* to the received trips, defined as Eq. 3.3, Eq. 3.4.

$$G = \frac{5}{7} \times (G_{0-12} + G_{0-24}) + \frac{2}{7} \times (G_{1-12} + G_{1-24}) \quad (3.4)$$

$$R = \frac{5}{7} \times (R_{0-12} + R_{0-24}) + \frac{2}{7} \times (R_{1-12} + R_{1-24}) \quad (3.5)$$

where the first sub-index corresponds to the weekday (0 for working days, 1 for weekends) and the second one corresponds to the first 12 hours of the day (12) or the last 12 hours (24).

Once the distance and variables used are defined, the other input needed is the number of clusters. To determine which number is more optimal the elbow method is used. That is, find the number of clusters K where the error curve has an inflection, which represents that the underlying model fits best with that parameter.

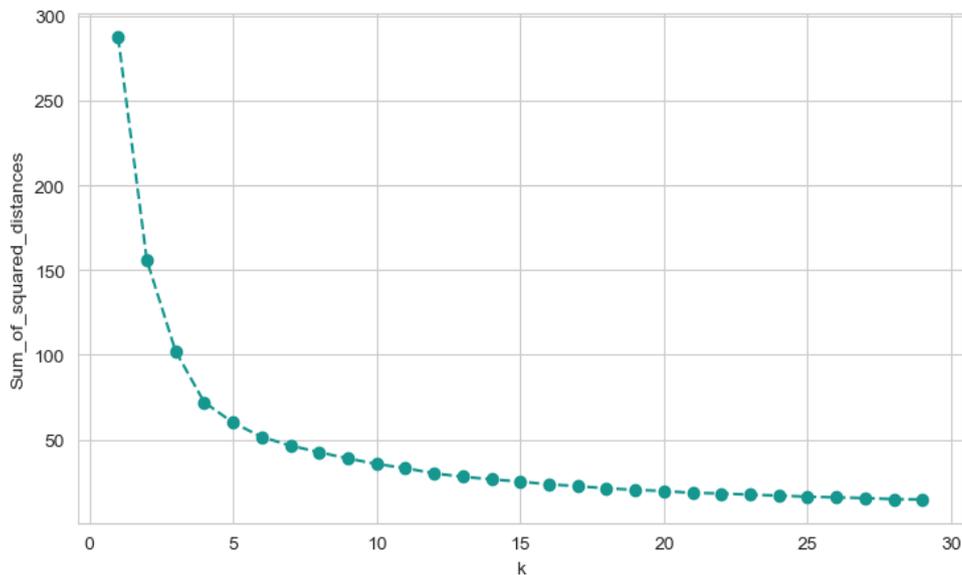


Fig. 3.7 Elbow method used to find the optimal number of clusters (K).

From Fig. 3.7, we actually see that in some cases, more than an optimal number we might find an optimal region of (in the figure above, between 4 and 6). In this particular case, 5 clusters are taken as the optimal number (Fig. 3.8).

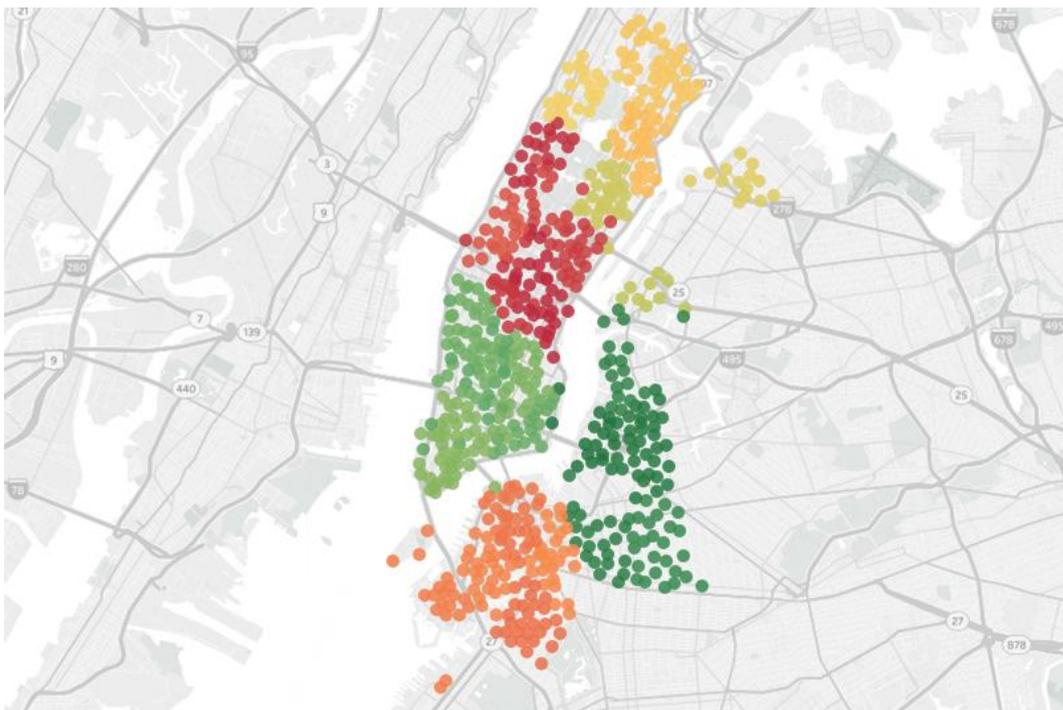


Fig. 3.8 Cluster arrangement in New York City with optimal $K = 5$.

It is still too computationally expensive to make a model for each of the clusters processed since their size is too big (in case of New York City around 130 stations per cluster). Therefore, the same method before is applied again to each one of the clusters to sub-divide them. This value is saved as $K_{main} \cdot K_{sub}$ in the database, which means that a cluster value of 1.2 means the station belongs to the cluster $K_{main} = 1$ and sub-cluster $K_{sub} = 2$ (Fig. 3.9)



Fig. 3.9 Example of a sub-cluster in New York City.

3.3.3 Effects of clustering in prediction accuracy

To determine whether the clustering technique applied is successful in minimizing the prediction error, and therefore increasing the accuracy, different cases have been analyzed in the demand prediction of a particular station. These cases are:

- Demand prediction taking only into account the single station being analyzed.
- Demand prediction taking into account j stations belonging to the same cluster.

Besides that, two different kind of stations have been studied, ones with high demand (HD) and low demand (LD), where demand can be defined as the sum of the total generated trips and received trips in the analyzed time period.

To obtain the wanted results, five different stations from what was considered a High Demand Cluster were analyzed and the results averaged. This was done for the study case of New York City and London City. Similarly, this was analyzed for Low Demand Clusters.

Since we will be considering fewer number of stations than the total available in the selected cluster, the test has been carried out five times for each station analyzed, randomizing the selection and then averaging the results. The accuracy in the model is calculated as Eq. 4.6

$$accuracy = 100 \times \left(1 - \frac{\sum(|x - x'|)}{\sum x} \right) \quad (3.6)$$

where x is the real observed value and x' is the predicted one.

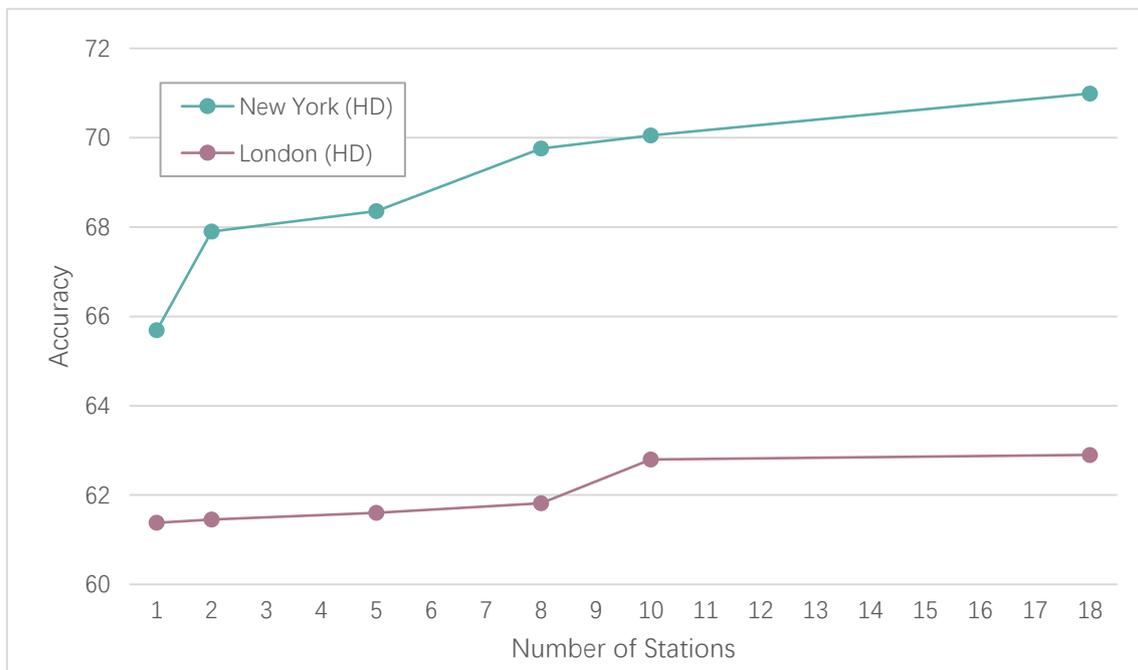


Fig. 3.10 Accuracy results for HD stations.

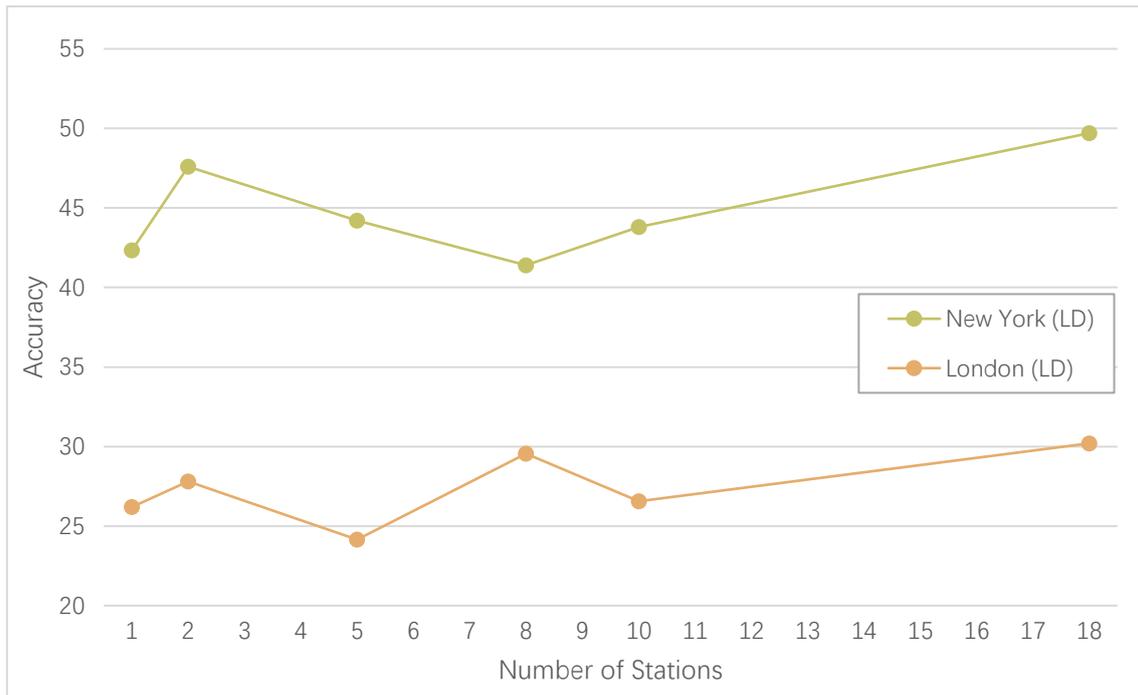


Fig. 3.11 Accuracy results for LD stations.

From Fig. 3.10 we can observe that for HD stations, the accuracy increases with the number of stations trained together. In Fig. 3.11, the accuracy does not always increase with number of stations, even though considering the total number of stations in the cluster seems to yield better accuracy than training a different model for each station.

Looking closer to the analysis carried out for a single station, there is a clear difference between HD and LD stations. The first ones have a much more linear behavior considering different station combinations, which is explained by the similarity in their overall demand patterns (Fig. 3.12). On the other hand, LD clusters have much more variability regarding different station combinations, since an overall lower demand is not enough to explain the time-variance of each individual station (Fig. 3.13).

It should also be considered that the time-step used for LD clusters was 2h, compared to the 1h for HD stations, since time-variability is too great for LD stations to carry out such fast-step analysis.

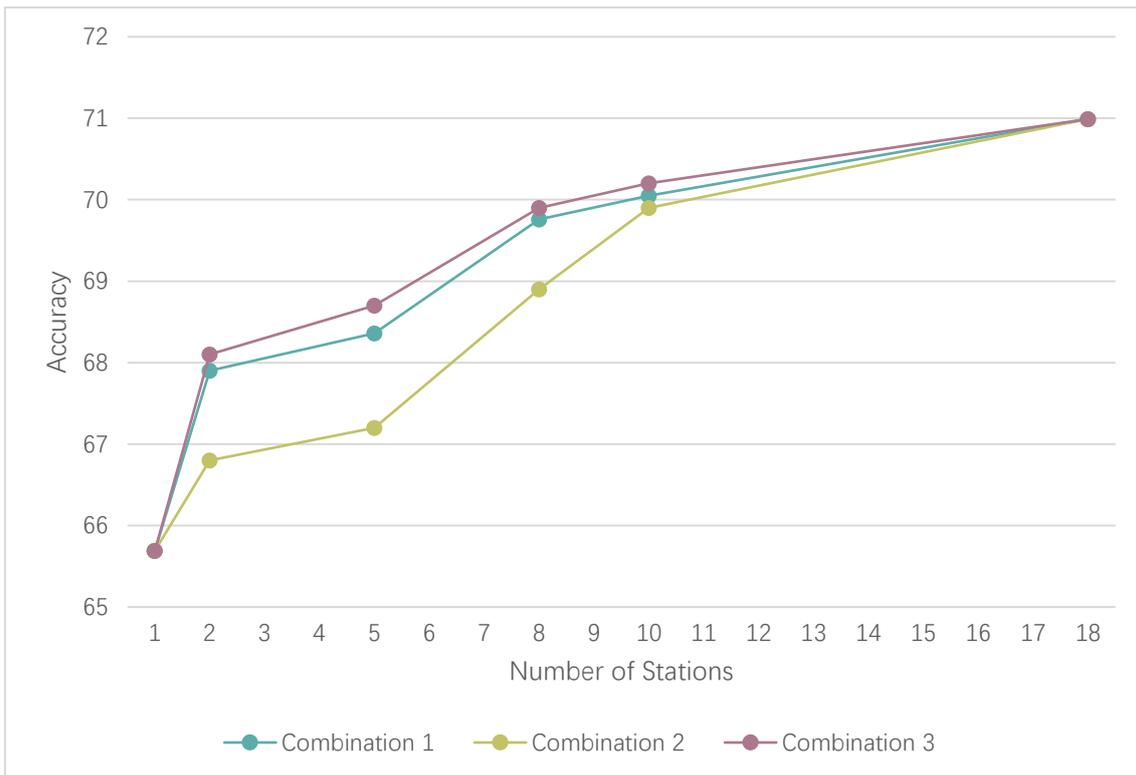


Fig. 3.12 Different combinations for HD stations prediction.

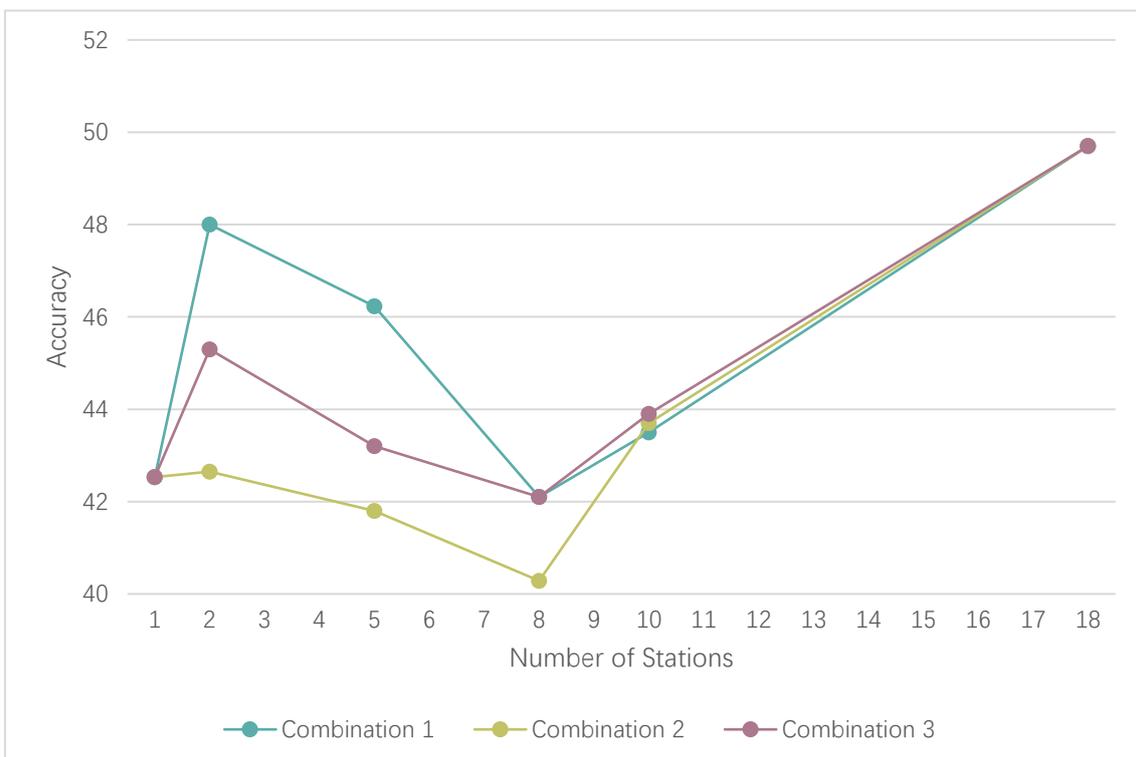


Fig. 3.13 Different combinations for LD stations prediction.

3.3.4 Clustering conclusions

We can conclude that clustering is a good approach to organize the great number of stations available for analysis, with a clear difference in HD and LD clusters. One could consider less stations, since there might be a good trade-off between computational time and accuracy, but at the end one should have a model for every station, therefore the number of stations to train in each model is considered to be the total number of stations in each of the sub-clusters.

Another feasible approach, also explored in this report, is to predict the overall demand levels of clusters, especially in LD clusters where the accuracy is generally lower. With this approach one would be able to predict, with more accuracy, the aggregated demand levels of a group of stations spatially close to each other.

3.4 Tested Algorithms

3.4.1 Random Forest Regression

The parameters considered to be optimized are:

- Number of estimators (*N_estimators*): This corresponds to the number of trees in the forest and the basic tuning parameter. The more trees the more accurate the model, but more computational time is taken.
- Max depth (*max_depth*): The maximum number of node-splits the algorithm will consider.
- Number of samples to split (*min_split_samples*): Minimum number of samples in each internal node for splitting to occur.
- Number of samples to stop (*min_samples_leaf*): Similar to the *min_split_samples*, but this acts on the leaf nodes rather than the internal nodes.
- Maximum number of features (*max_features*): The maximum number of features to consider when you are splitting a node.

From Fig. 3.14 we can see that with only 100 trees we already reach the maximum accuracy with less of 2.5s of training. This calculation was done for a high – demand cluster but similar results are obtained for a low – demand one, the only differences being lower computational time and lower accuracy.

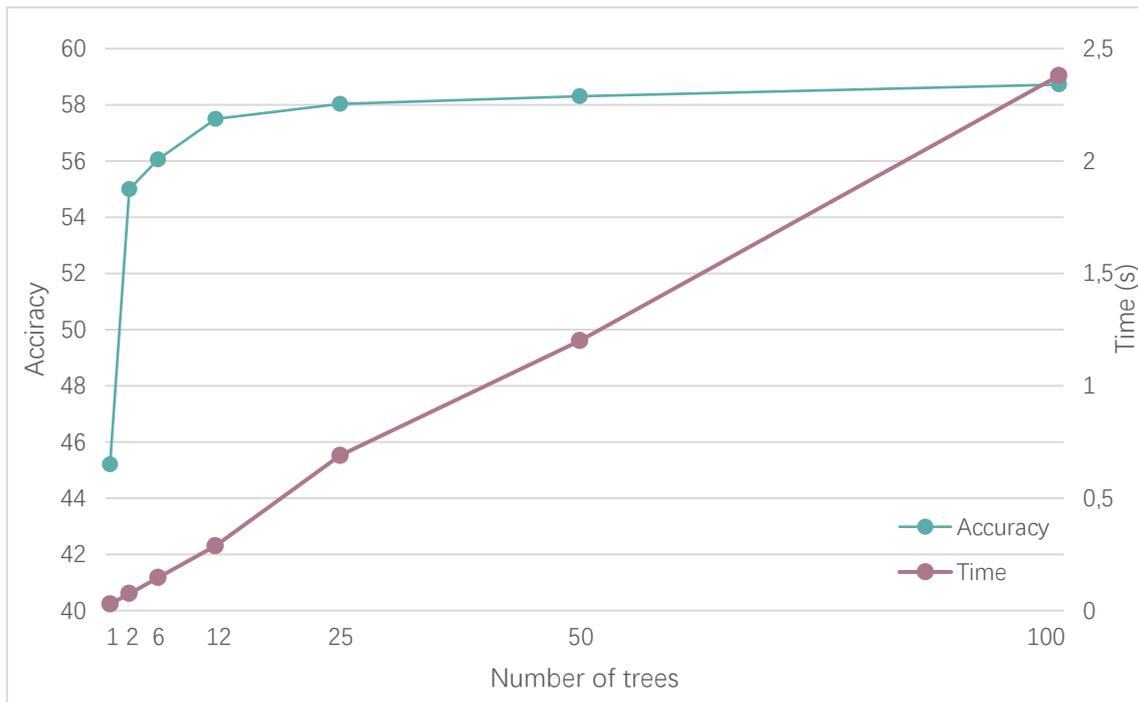


Fig. 3.14 Accuracy results VS number of trees in the RF.

From the other parameters to optimize, a Grid Search was carried out, with the results being that *max_features* should be left to *auto*, *max_depth* without any conclusive results, *min_samples_leaf* equal to 4 and *min_samples_split* equal to 2.

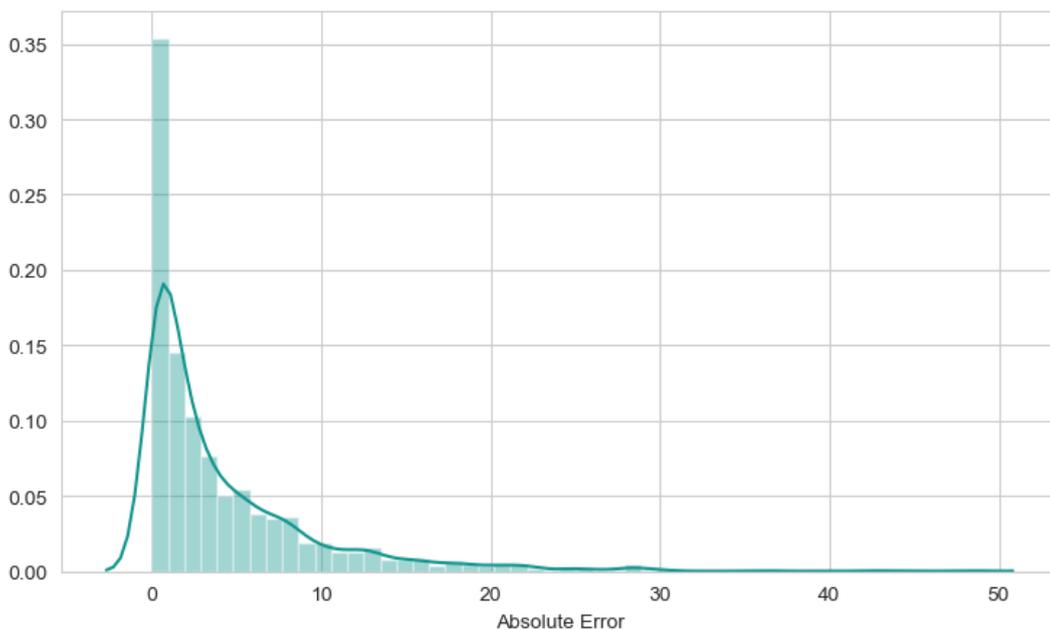


Fig. 3.15 Error distribution of the optimized Random Forest.

3.4.2 Gradient Boosting Method

The parameters to tune are the same ones than a Random Forest, since both methods are ensemble algorithms. The main difference is that there is a learning rate parameter to tune, which shrinks the contribution of each tree (Fig. 3.16).

There actually is a trade-off between the number of estimators and the learning rate, since the more trees and the higher the learning rate, there are more possibilities to overfit. Thus, a Grid Search based on these two parameters is carried out (Table 3.2).

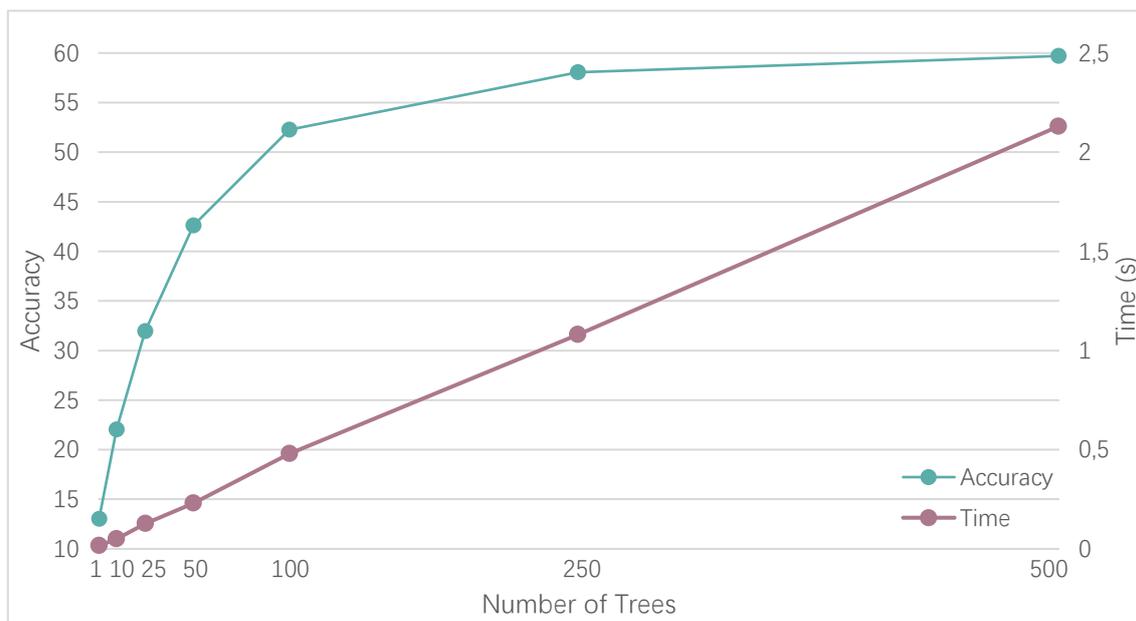


Fig. 3.16 Accuracy results VS number of estimators in GBM.

Table 3.2 Grid Search for the optimal parameters for the GBM.

Trees	10	100	500	1000	2000	
Time (s)	0,048	0,447	2,44	5,00	10,15	
Learning Rate	0,01	12,94	21,56	41,76	51,34	56,42
	0,05	17,67	42,03	57,53	59,35	60,49
	0,1	22,04	52,27	59,70	60,70	60,78
	0,15	25,89	55,45	60,46	60,75	59,93
	0,2	29,24	56,88	61,06	60,85	59,97
	0,25	32,21	57,60	60,84	60,19	59,18

From Fig. Fig. 3.17, we can see that generally speaking, increasing the learning rate and number of estimators does increase the accuracy. Having said that, there is a limit for when the model overfits, where the accuracy actually decreases. From the obtained data, the optimal number of estimators is supposed to be 500, whereas the learning rate is equal to 0,2. The minimum number of samples in leaf node is 2, as well as the minimum samples to split in an internal node.

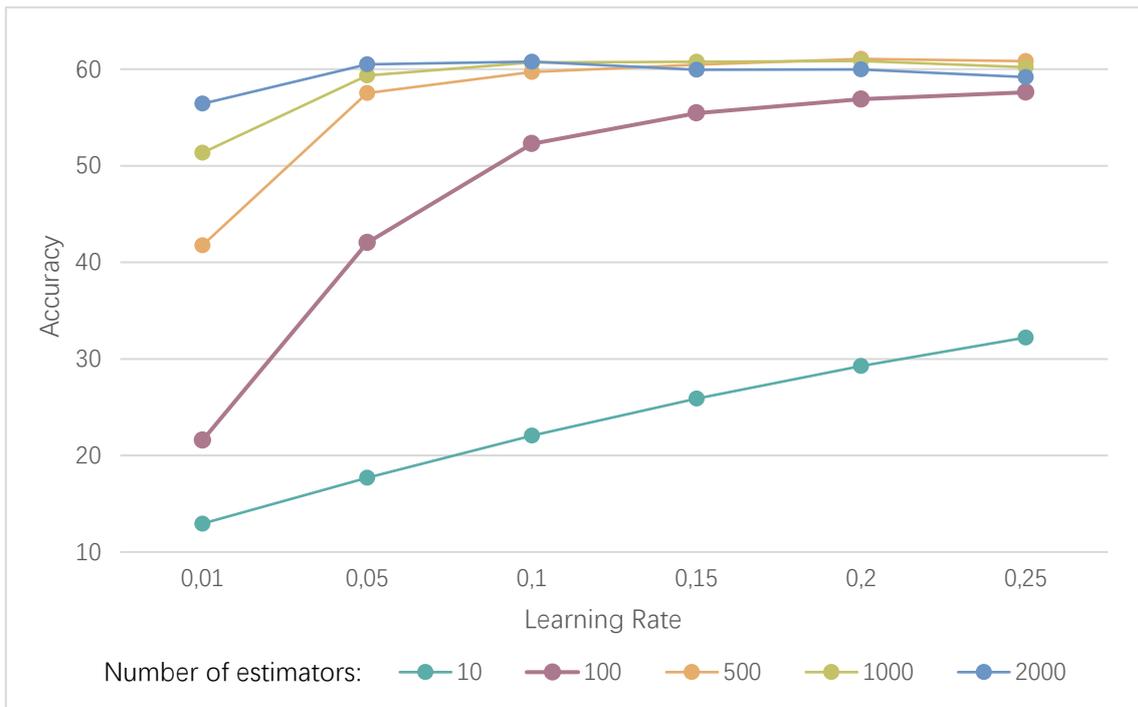


Fig. 3.17 Accuracy VS Learning rate for different number of estimators.

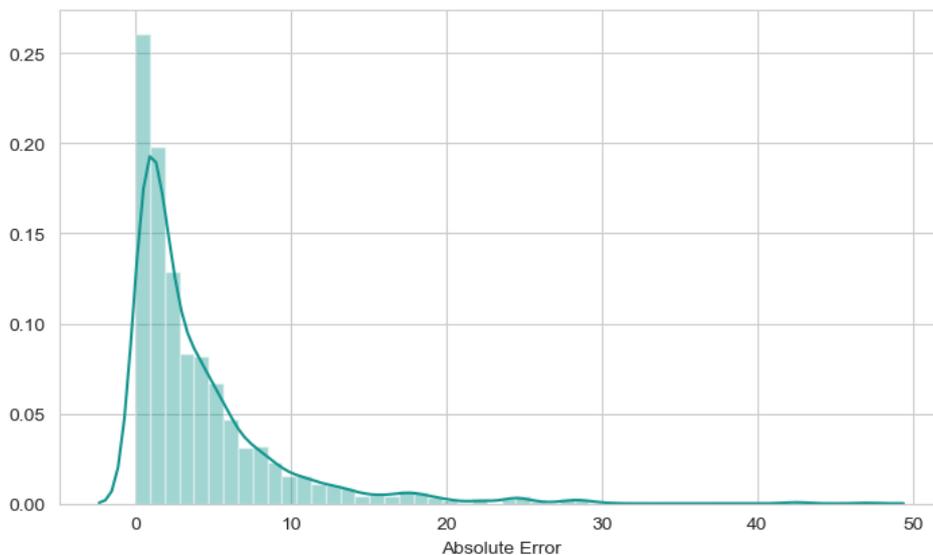


Fig. 3.18 Error distribution of the Gradient Boosting Regressor.

3.4.3 Deep Neural Networks

Unlike Random Forest or Gradient Boosting, parameter tuning in NN also includes finding the optimal structure of the algorithm. Therefore, different layouts and activation functions have been tested to see which yields the best accuracy with the lowest computational time possible. To consider the “optimal” layout in each case, the number of epochs has been monitored to be the less possible without underfitting (Table 3.3).

The loss function to calculate the error between the output value and the real value is defined as the mean squared error (Eq. 3.7):

$$MSE = \frac{1}{n} \sum_{i=1}^n (X_i - X_i')^2 \tag{3.7}$$

where n corresponds to the number of data points, X_i the real value and X_i' the predicted one.

Table 3.3 Structure arrangements for the NN which have been tested.

Layout	Parameters	Input	Hidden 1	Hidden 2	Hidden 3	Output
Layout 1	Neurons	n_inputs	n_inputs	-	-	1
	Act. Func.	-	tanh	-	-	relu
Layout 2	Neurons	n_inputs	2×n_inputs	-	-	1
	Act. Func.	-	tanh	-	-	relu
Layout 3	Neurons	n_inputs	n_inputs	n_inputs/2	-	1
	Act. Func.	-	tanh	relu	-	linear
Layout 4	Neurons	n_inputs	n_inputs	n_inputs	-	1
	Act. Func.	-	tanh	relu	-	linear
Layout 5	Neurons	n_inputs	2×n_inputs	n_inputs	-	1
	Act. Func.	-	tanh	relu	-	linear
Layout 6	Neurons	n_inputs	2×n_inputs	n_inputs/2	-	1
	Act. Func.	-	tanh	relu	-	linear
Layout 7	neurons	n_inputs	n_inputs	n_inputs	n_inputs	1
	Act. Func.	-	tanh	relu	relu	linear
Layout 8	Neurons	n_inputs	2×n_inputs	n_inputs	n_inputs/2	1
	Act. Func.	-	tanh	relu	relu	linear
Layout 9	Neurons	n_inputs	n_inputs	n_inputs/2	n_inputs/4	1
	Act. Func.	-	tanh	relu	relu	linear
Layout 10	Neurons	n_inputs	n_inputs	2×n_inputs	n_inputs	1
	Act. Func.	-	tanh	relu	relu	linear

Table 3.4 Accuracy and computational time results for different layouts.

Layout	Epochs	Time	Accuracy
Layout 1	25	2,983	24,75
Layout 2	25	3,107	25,24
Layout 3	50	5,842	61,52
Layout 4	50	6,523	61,32
Layout 5	50	7,321	61,15
Layout 6	50	7,656	61,05
Layout 7	30	6,342	60,98
Layout 8	30	6,316	60,76
Layout 9	40	7,944	50,54
Layout 10	30	7,457	61,54

We can see from Table 3.4 that just one layer (layout 1 and layout 2) is not enough to train our model, but two and three layers seem to predict quite well the output. Accuracy wise, Layouts 3, 4, 5 and 10 outperform the others. Between these, the one with less computational time is Layout 3, with only 5,8 seconds to build the model, whereas Layout 6 takes the longest, with 7,65 seconds.

Even though the output considered initially was one neuron, corresponding to the bike demand, the final model integrates two output neurons, corresponding to both In and Out demand. Thus, a single model will be able to predict both demands at the same time.

From the results, we can conclude that Layout 3 is the optimal one for our problem and therefore used to build up the models when Neural Networks are the method of use.

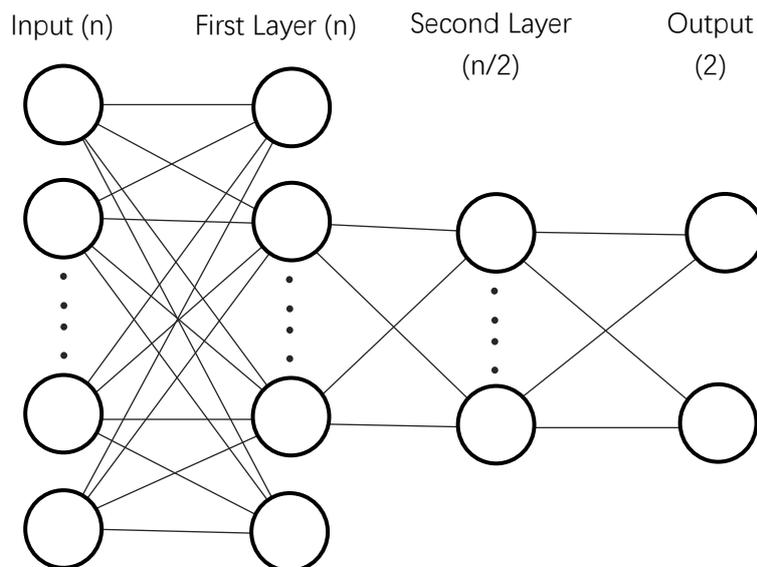


Fig. 3.20 Scheme of the layout used for the Neural Network analysis (Layout 3).

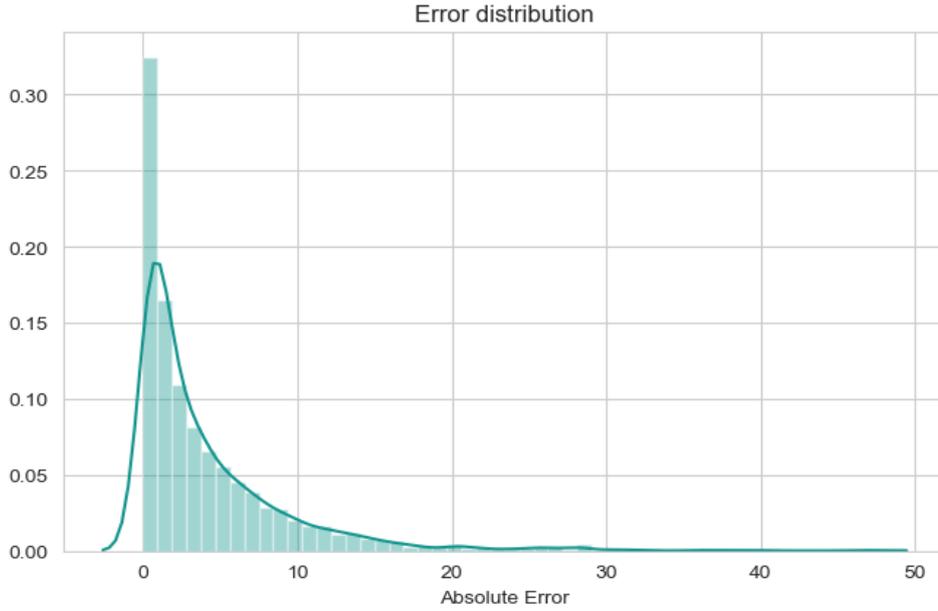


Fig. 3.21 Error distribution of the Deep Neural Network.

3.4.4 Model comparisons

To compare the behavior of the analyzed algorithms, different error functions are used. Besides accuracy, RMSLE (Root Mean Square Logarithmic Error) (Eq. 3.8) and RMSE (Root Mean Square Error) (Eq. 3.9) are used to be able to compare the results with other studies, where these error definitions are often used.

$$\text{RMSLE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(X_i + 1) - \log(X'_i + 1))^2} \quad (3.8)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (X_i - X'_i)^2} \quad (3.9)$$

where n corresponds to the number of data points, X_i the real value and X'_i the predicted one.

RMSLE is usually used when you do not want to penalize large differences when the predicted and real values are large, since it considers the percentual difference rather than the absolute one. On the other hand, RMSE can be understood as the standard deviation of the residuals.

Table 3.5 Overall results of the different tested algorithms.

	Time (s)	Acc. (%)	Avg. Error	RMSLE	RMSE
Random Forest	4,76	58,72	4.15	0,54	0,23
Gradient Boosting	4,88	61,06	4,08	0,51	0,23
Neural Network	5,84	61,52	4,07	0,51	0,23

Comparing the three optimized algorithms, we can see that the Neural Network has the highest accuracy, whereas Random Forest has the lowest one. As for computational time, NN takes longer than both other models. Thus, GBM seems to have the best accuracy – computational time relation. It should be noted that the computational time considered is the one taken to build the model, which is just done once. Besides that, the NN layout is able to predict both outputs at the same time, whereas RF and GBM build two completely different models to be able to output two different variables.

Having said that, the three algorithms will be used in the following case studies to further compare their performance, since not only they need high accuracy and low computational time, but they need to be able to adapt to the different analyzed cases.

Chapter 4: Case Studies

In this Chapter, two case studies will be presented corresponding to two different cities: New York and London. These two systems in particular have been chosen over others because they provide users with a greater amount of data (more users use their system) as well as well-structured registers. For both cases, the data the bike-sharing system of 2018 and part of 2019 was extracted, in addition to the historical weather for those dates and places. All this data was treated following the steps of part 3.1 (standardization, discretization...).

Using the data of 2018 as training and testing set, the extracted part of 2019 (January) was used as validation data, therefore the presented results in this Chapter will correspond to those dates. As mentioned before, the validation test is part of the dataset which the algorithms have never seen before, an actual prediction, with the only difference being that you already know the outputs, therefore being able to calculate the errors.

For each case study, the available data of the analyzed system will first be presented, including time and spatial distribution of the demand, as well as the final discretization intervals of the weather conditions particular for the city. Then the carried-out predictions will be analyzed, which include a city-level, a cluster-level and a station level prediction. Two kind of errors will be analyzed, the first one being the test-set and the other the validation set. The first one provides more general information about the behavior of the algorithm, whereas the second one can be further analyzed, being able to discern between different types of days (working days, holiday) or different weather conditions. Additional analysis will be done concerning the effect of different time-steps and how the inherent error in weather forecast might increase the prediction error.

4.1 New York

New York Citi Bike system opened in May 2013 with 322 stations at first, with added up the total number of 6.000 bikes in their system. There have been several expansions, which have reached the final number of 706 stations and 12.000 bikes by October 2017 [43].

There are two type of users, differentiated between short-time pass users and annual pass users. The first one is more catered two tourists, with a single ride for \$3, a 24h pass at \$12 and a 3-day one at \$24. In every case, the trip is limited to 30min, with an extra fee of \$4 for each additional 15min. The second one consists on a single flat fee of \$169, with the trips limited to 45min, having an extra fee of \$2.5 for any additional 15min.

4.1.1 Data exploring

4.1.1.1 Weather

The weather conditions in the city are discretized based on their quantile distribution, with the results in Table 4.1. It should be noted that for rain and snow, the first category (category 0) should be understood as lack of precipitation and hence it contains more samples than the other categories for the same variable. For snow in particular, only two categories are considered, corresponding two whether it is snowing (binary variable).

Table 4.1 Weather conditions discretization.

Category	Temperature (°C)		Wind (m/s)		Rain (mm)		Snow (cm)	
	L _{Boundary}	U _{Boundary}						
0	-	1,83	-	1,604	-	0,001	-	0,001
1	1,83	6,83	1,604	3,208	0,001	0,061	0,001	50
2	6,83	15,17	3,208	4,812	0,061	0,259	-	-
3	15,17	22,42	4,812	6,416	0,259	0,962	-	-
4	22,42	-	6,416	-	0,962	-	-	-

4.1.1.2 Demand behavior

Before carrying out any kind of prediction, it is necessary to understand how the system behaves in order to be able to reach accurate conclusions. This is done by calculating the number of bikes being used in the city depending on different variables, such as day of the week or season.

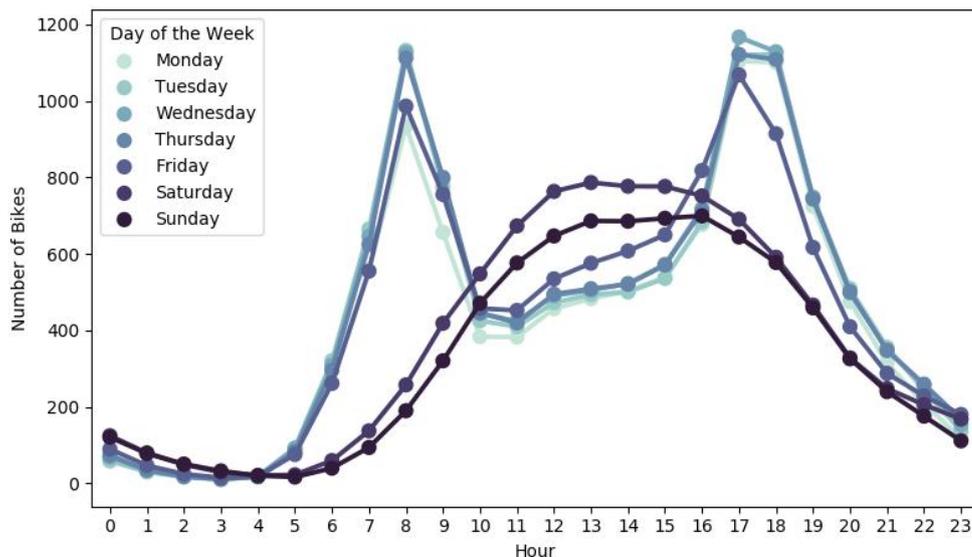


Fig. 4.5 Distribution of the bikes in use depending on the weekday.

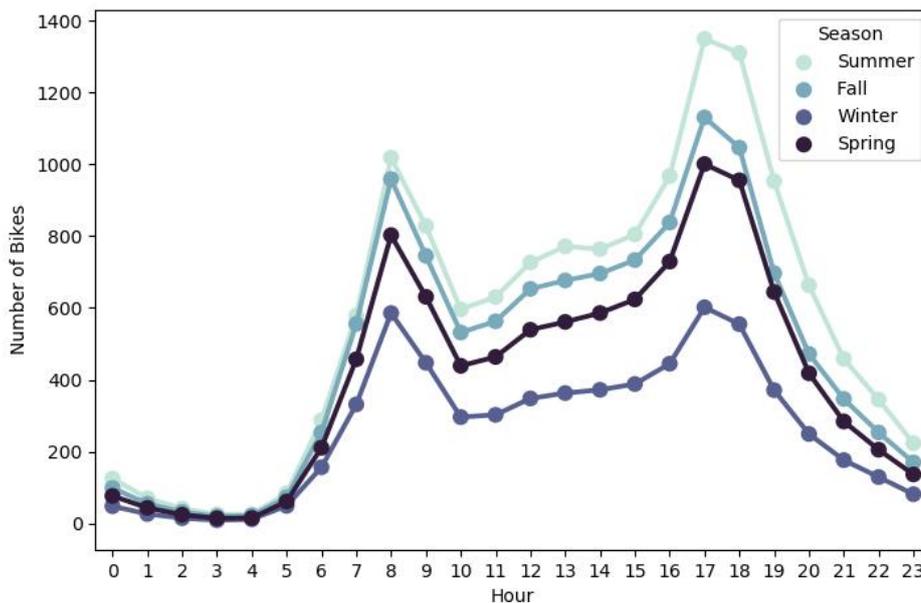


Fig. 4.6 Distribution of the bikes in use depending on the season.

As seen in 0, the distribution of the bikes during the day heavily depends on whether it is working day or weekend. For working days, there are two clear peaks in demand, the first one around 8am and the second one between 5 and 6pm. On the other hand, weekends follow a smoother distribution, reaching its maximum value around midday. It should be noted that, in this case, Friday follows almost exactly the same distribution as any other working day, which is due to the fact there is not such a thing as “intensive schedule” (Friday afternoon off) in the city.

In Fig. 4.6, we can observe that the overall highest demand is in Summer, whereas the lowest one is in Winter. This is mainly explained by the effect of the temperature, since in Winter it might be too cold to regularly use bikes. The higher use of bikes in summer might be explained by the increased number in tourists using the bike system and the lack of extensive working holidays.

4.1.1.3 Spatial behavior

It also interesting to see how these demand patterns are distributed around the city. First, the number of trips generated and received for each station during a year are plot against the city map. From here, we are able to tell which is the center of demand in the city and wether the generating and receiving areas are the same or the system is decompensated.

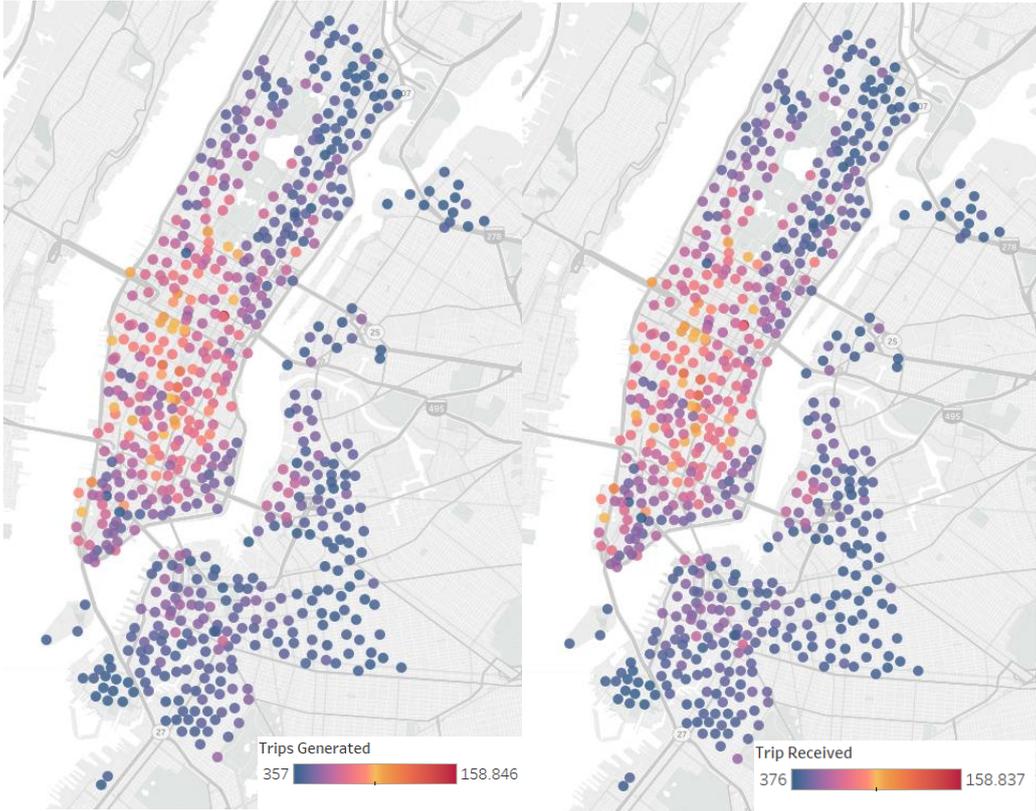


Fig. 4.7 Trips generated (left) and trips received (right) in New York during a year.

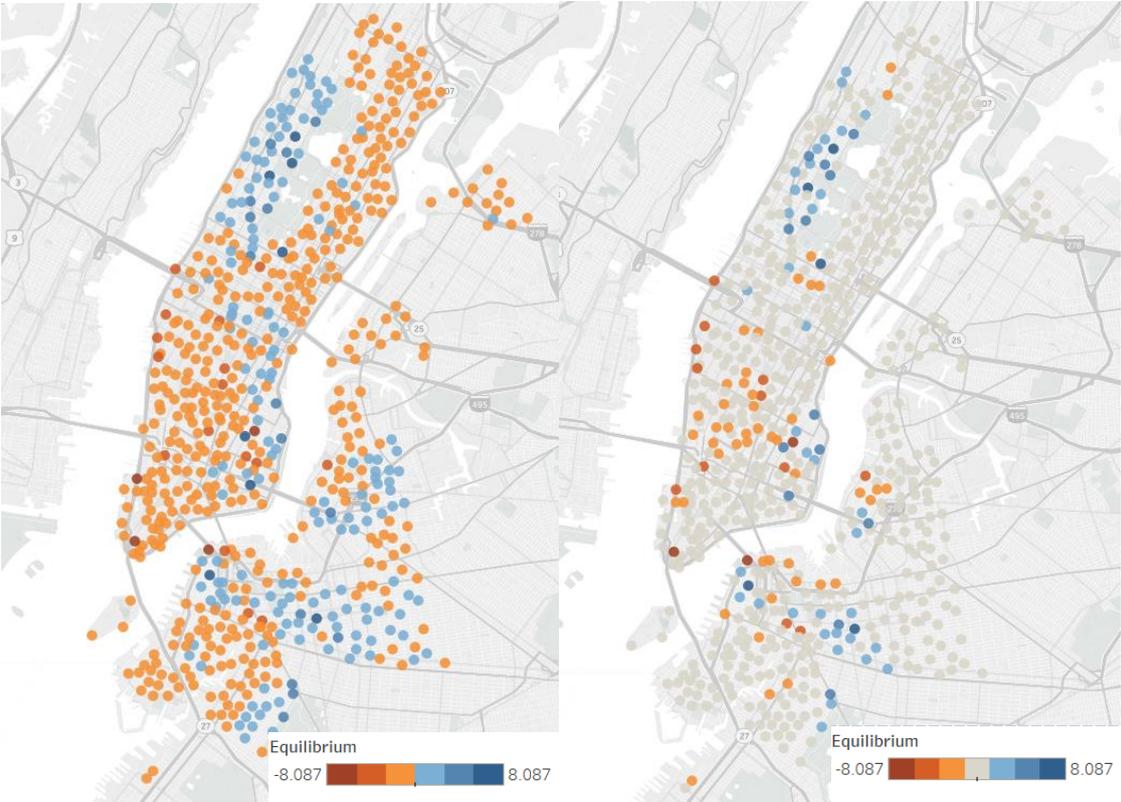


Fig. 4.8 System equilibrium (trips received minus trips generated).

From Fig. 4.7 we appreciate that the center of demand, for both generated trips and received trips, appears to be in Midtown Manhattan, which corresponds of the main office area of the city, whereas there is less demand outskirts of the city center. This is an important factor to take into consideration, since stations with less demand are less predictable, and larger time-steps are needed to reach the same accuracy as high-demand stations.

To see the imbalance in generated and trips received, the difference between these two is also plot. From Fig. 4.8 (left), one can appreciate that the stations outskirts of the city tend to generate more trips than receive (positive equilibrium), whereas the center receives more. Overall, most of the stations are in an equilibrium closer to zero (Fig. 4.8right). Thus, the possible imbalance in the left part of Fig. 4.8 is mainly due to one-way trips from the outskirts to the center, even though they do not represent the majority of the trips between these two zones. The center part of the city also corresponds to the area where the stations with higher capacity are situated (Fig. 4.5).

Finally, it is also interesting to try to distinguish different motives for the ride. This can be partially done by evaluation the average trip duration of all the trips in a certain station (Fig. 4.6). From there, it is easy to identify certain routes and stations that are more used for touristic or exercise purposes rather than commuting, since they tend to have larger average trip durations.

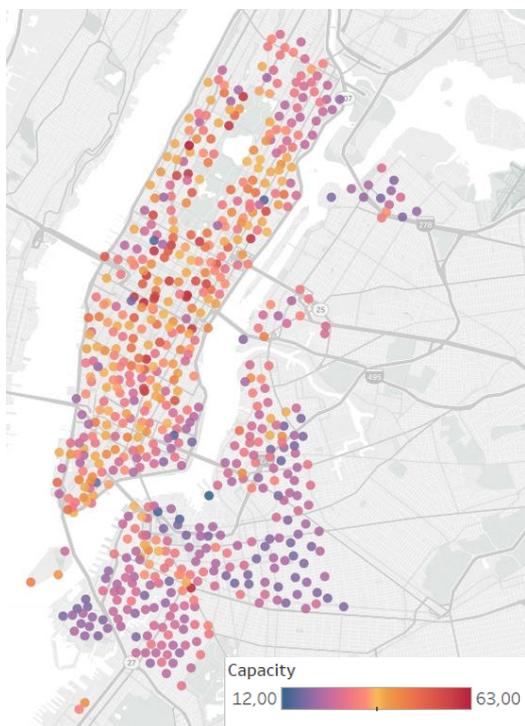


Fig. 4.5 Capacity distribution of the stations in New York City.

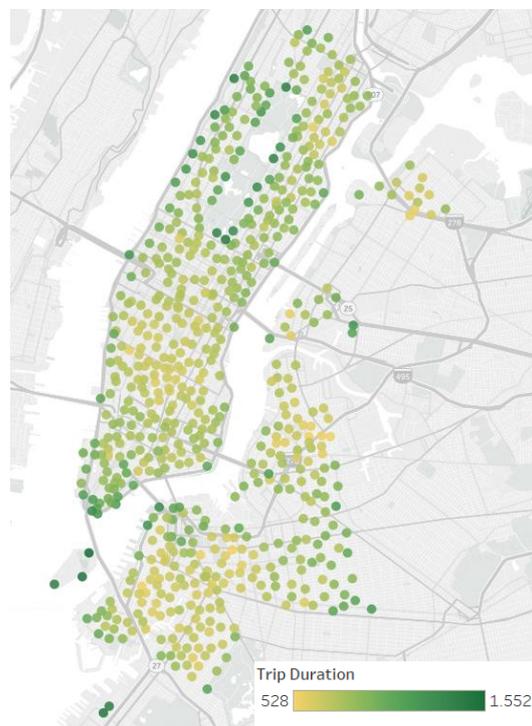


Fig. 4.6 Trip duration distribution of the stations in New York City.

4.1.2 City – level prediction

The three proposed algorithms in Chapter 3 are trained with the data of 2018 to predict 2019. Since the demand is aggregated for every station, the chosen time-step is just 15min, which is enough to reach an acceptable accuracy level.

The precision of the prediction varies depending on the particular dates that are predicted, the error results considered correspond to the test set, corresponding to the 20% of the input data (with training set corresponding to the 80%). As for the predicted variable, both In and Out demand is predicted at the same time, with the presented error being the average of the error in the prediction of the demand in the station and the demand out of it.

Table 4.2 Overall precision results for a city-level prediction of New York City.

	Time (s)	Acc. (%)	Avg. Error	Expl. Var	Max. Error	RMSLE	RMSE
Random Forest	19,07	77,31	96,985	0,834	1284,28	0,186	0,413
Gradient Boosting	24,79	74,32	109,77	0,824	1191,22	0,189	0,425
Neural Network	144,23	79,64	87,020	0,883	1310,155	0,168	0,346

* Average number of bikes equal to 427,43 bikes.

From Table 4.2 we can see that the best accuracy is reached with a Neural Network (79,64%) even though Random Forest comes close with a 77,31% of accuracy. As for the explained variability, NN can explain 88,3% of it, whereas RF has a value of 83,4% and Gradient Boosting 82,4%. Even though NN has the highest accuracy, it also has the maximum error (1310,155), whereas GBM has the lowest one with 1191,22 bikes.

Again, it should be noted that NN takes around 10 times longer to build a predictive model compared to a Random Forest. Even though this is only done once, since the model is then saved, it should be taken into consideration if a lot of different models have to be built.

Once the errors for the validation test are analyzed, one can start to focus on the behavior of the algorithms with completely new data. Any possible dates of 2019 could be predicted, but the last two weeks of January are chosen since they have most of the different studied cases, such as holiday, rainy days or weekends. Even though both demands generated and received are predicted, the figures showcase the particular case of demand received, since both cases are too similar in a city-level prediction.

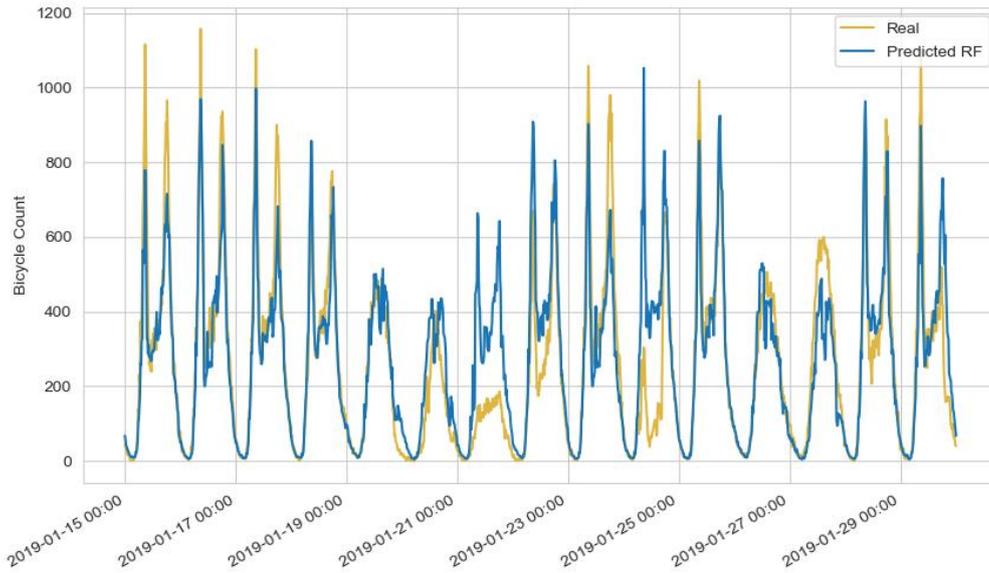


Fig. 4.7 City – level demand prediction from 15/01/2019 to 29/01/2019 using Random Forest.

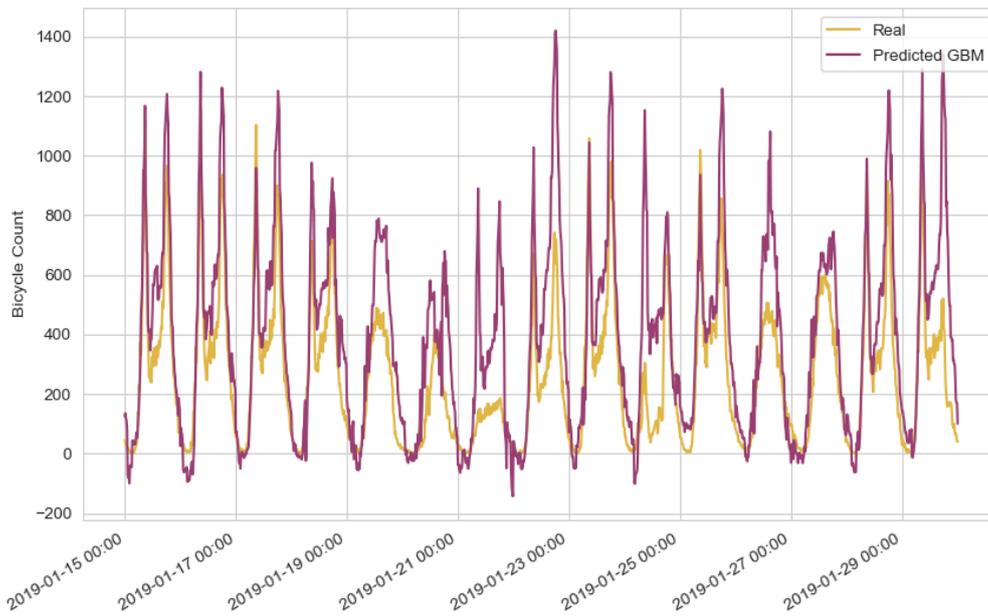


Fig. 4.8 City – level demand prediction from 15/01/2019 to 29/01/2019 using Gradient Boosting.

Random Forest seems to adjust quite well to the prediction in most cases but it falls short when it is time to predict the peak-hour demand (Fig. 4.7). There seem to be some extra errors on the 21st of January, since it was holiday, even though the algorithm knew that Monday was not a working day. The predicted demand for that day is lower than other Mondays, but it seems it should be even lower. Some extra errors also occur on the 24th, where the collected weather data mentioned it did not rain, even though some other sources say the opposite. It could be possible that the particular used weather station did not collect any precipitation, but it did in fact rain, which would explain the lower demand that day.

Gradient Boosting seems to reach the peak-hour demand better than RF, even surpassing its value most of the days and predicting an overall higher demand on the weekends (Fig. 4.8). Again, extra errors on the 21st and 24th due to national holiday and rain respectively. It is interesting to see that for the non-peak hours at night, the algorithm actually predicts a negative bike count value. This could be tweaked in the future, making all the negative predicted values equal to zero, but to fairly compare the different algorithms precision, this was not done in this case.

Finally, the prediction using Neural Networks yields similar conclusions (Fig. 4.9). It should be noted that for the particular days 21st and 24th of January, it seems to predict the values quite better than the other algorithms, which yields to lower maximum error. On the other hand, peak values seem to be over-predicted, as well as non-peak ones. It is also interesting to see the computed confidence interval (95%), acceptable in most of the days but particularly wide in days with rains, holidays or even weekends.

All the error results are showcased in Table 4.3, dividing the results in normal workdays (no holidays and no rain), weekends and holidays (no rain), rainy (or snowy) days, peak hours (no rain) and overall results.

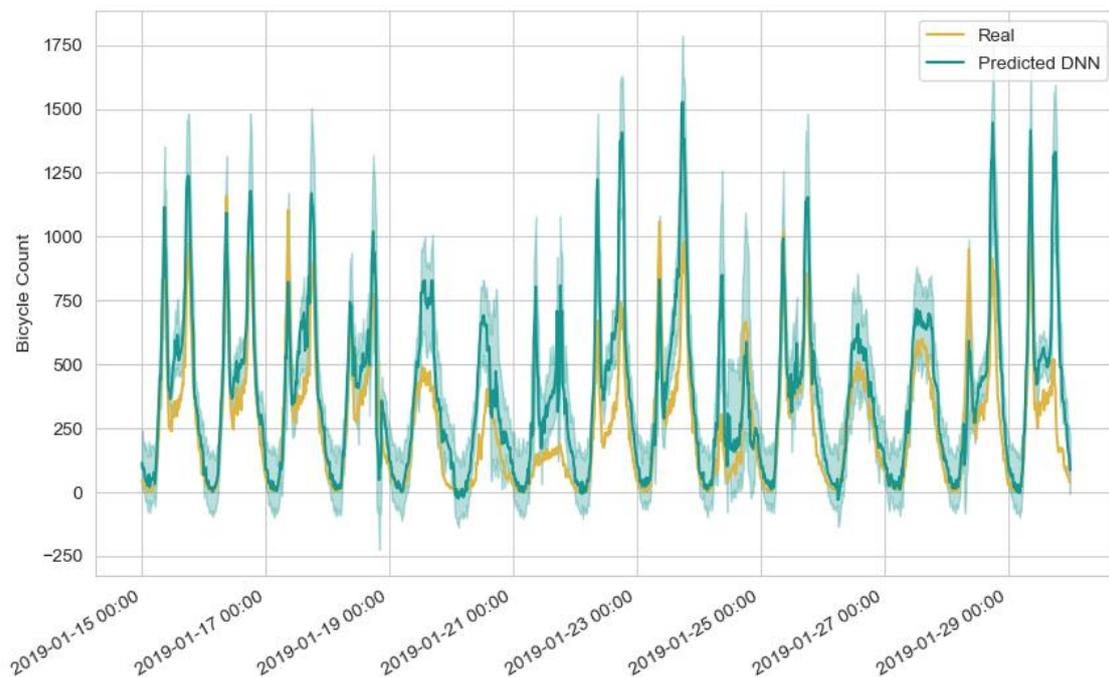


Fig. 4.9 City – level demand prediction from 15/01/2019 to 29/01/2019 using Neural Networks.

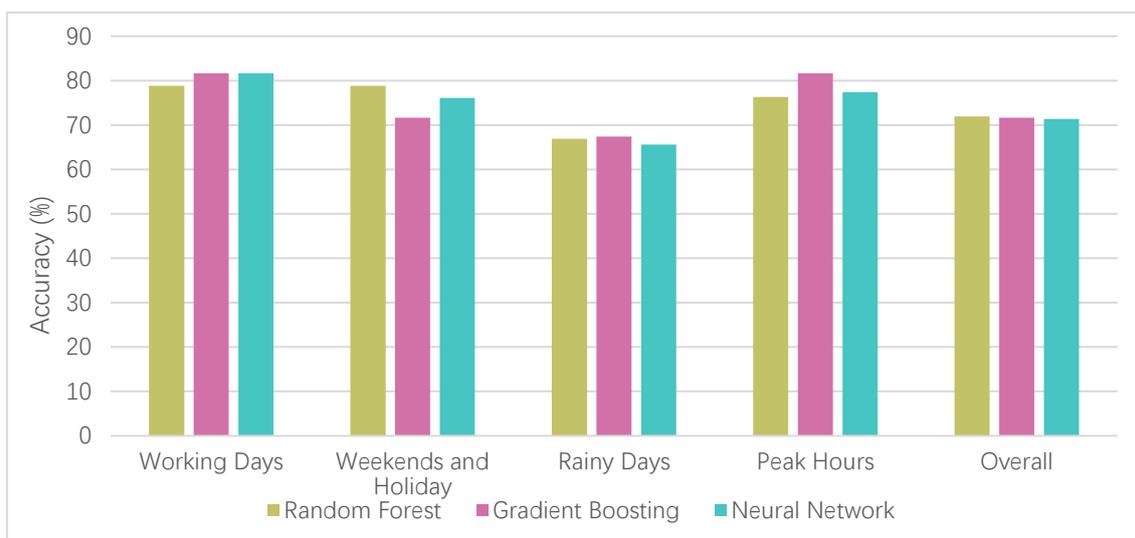


Fig. 4.10 City – level accuracy results from 15/01/2019 to 29/01/2019.

Table 4.3 City – level error results from 15/01/2019 to 29/01/2019.

		Random Forest	Gradient Boosting	Neural Network
Working Days	Avg. Bikes	319,114	319,114	319,114
	Avg. Error	67,447	58,576	58,678
	Acc. (%)	78,86	81,64	81,61
	Max. Error	420,251	442,011	450,852
Weekends and	Avg. Bikes	227,351	227,351	227,351
	Avg. Error	40,085	64,554	54,378
	Acc. (%)	78,85	71,61	76,08
	Max. Error	300,514	287,802	243,528
Rainy days	Avg. Bikes	252,009	252,009	252,009
	Avg. Error	83,387	82,231	86,702
	Acc. (%)	66,91	67,37	65,60
	Max. Error	748,284	727,304	615,540
Peak Hours	Avg. Bikes	739,233	739,233	739,233
	Avg. Error	175,129	135,88	167,015
	Acc. (%)	76,31	81,62	77,41
	Max. Error	748,284	727,304	473,587
Overall	Avg. Bikes	260,593	260,593	260,593
	Avg. Error	73,134	73,954	74,621
	Acc. (%)	71,93	71,61	71,36
	Max. Error	748,284	727,304	615,540

From the error results obtained from the prediction, one can see that Random Forest and Gradient Boosting get the best results, with NN just excelling in the maximum error computing. Overall, RF gets the lowest Avg. Error and highest accuracy (Fig. 4.10), followed closely by GB. In particular, it seems GB gets better results in peak hours, rainy days and working days, whereas RF is better in predicting weekends and holidays. It should be noted that by just fixing the negative values in the GB prediction, the overall results will exceed the RF in every aspect. Noticeably, the maximum error occurs for rainy days, which means the predictions during those days is highly unpredictable.

In conclusion, even though NN got the best results in the testing set (Table 4.2), that does not mean that it will yield the best precision in every prediction, as just proved by predicting these two particular weeks in January, where GB performs better than all the other algorithms tested. For example, by just predicting the first two weeks of January instead of the last two we obtain that RF yields overall better results (Table 4.4).

Table 4.4 City – level error results from 01/01/2019 to 15/01/2019.

		Random Forest	Gradient Boosting	Neural Network
Overall	Avg. Bikes	292,375	292,375	292,375
	Avg. Error	75,372	82,776	80,949
	Acc. (%)	74,22	71,69	72,31
	Max. Error	969,880	1015,843	1033,070
	RMSLE	0,149	0,155	0,157
	RMSE	0,330	0,333	0,346

It is also interesting to try to evaluate the difference between the two demands, that is demand into the system minus demand out of it, to try to exactly evaluate how many bikes are needed in the system at a particular time. To do that, the time-step is changed from 15min to 1h to reduce the variability of the result, since increasing the time-step increases the overall accuracy of the algorithms. This aggregation of the time-step is done to the already predicted values with a time-step of 15min, even though the same results are obtained if the initial trained time-step is increased. Thus, even though a small time-step has lower accuracy, it is used so this post-aggregation can be done.

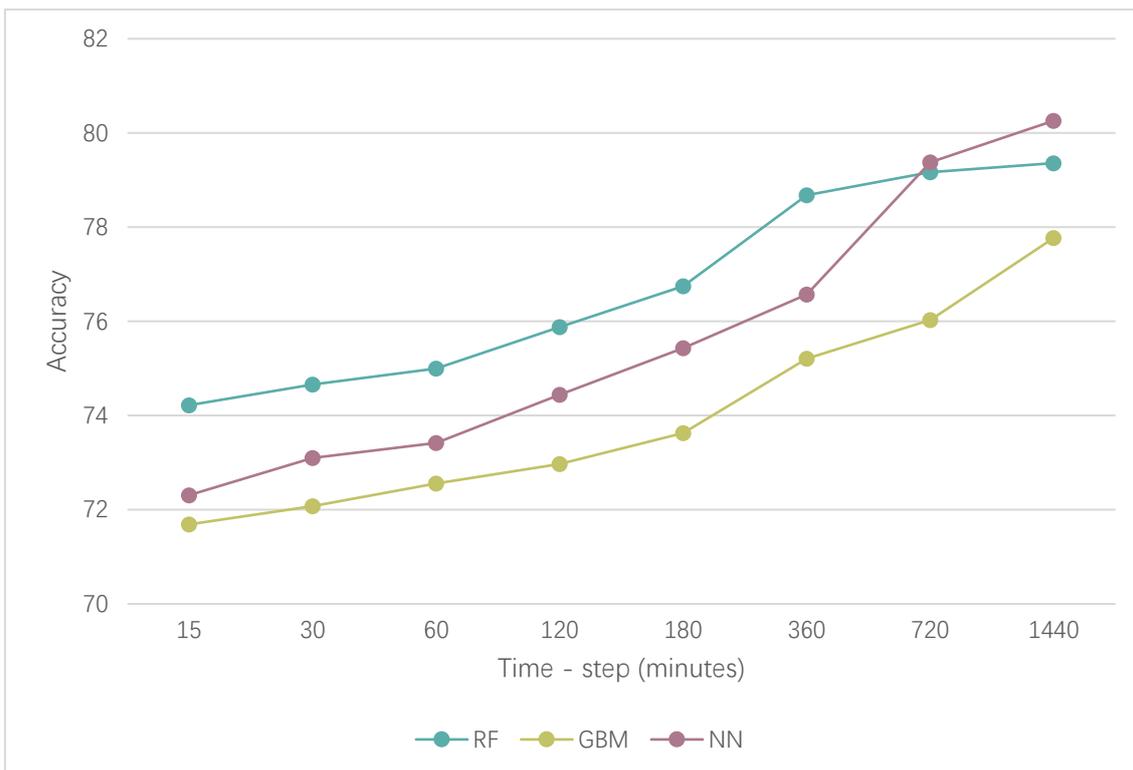


Fig. 4.11 Accuracy with increased time – step in the city-level prediction from 01/01/2019 to 15/01/2019.

Since the average accuracy increased for the prediction of the demand in and out of the system (Fig. 4.11), the accuracy yield from the subtraction from these two variables (equilibrium in the system) will also increase. Thus, a time-step of 360 min (six hours) is used to plot the results.

From the obtained results (Table 4.5, Fig. 4.12) we can observe that, even though the time-step has been increased, it is extremely difficult to yield a good precision in the prediction of the system’s equilibrium, reaching a maximum of 36,36% of accuracy using NN. This is due to the fact that, by evaluation the subtraction of the two demands predicted, we are increasing the variability in the result, which tremendously effects the overall prediction results.

Table 4.5 City – level error results from 01/01/2019 to 15/01/2019 (in – out demand).

	Random Forest	Gradient Boosting	Neural Network
Overall	Avg. Bikes	228,035	228,035
	Avg. Error	174,059	167,416
	Acc. (%)	23,67	26,58
	Max. Error	1073,279	1074,633
	RMSLE	0,342	0,335
	RMSE	0,889	0,761

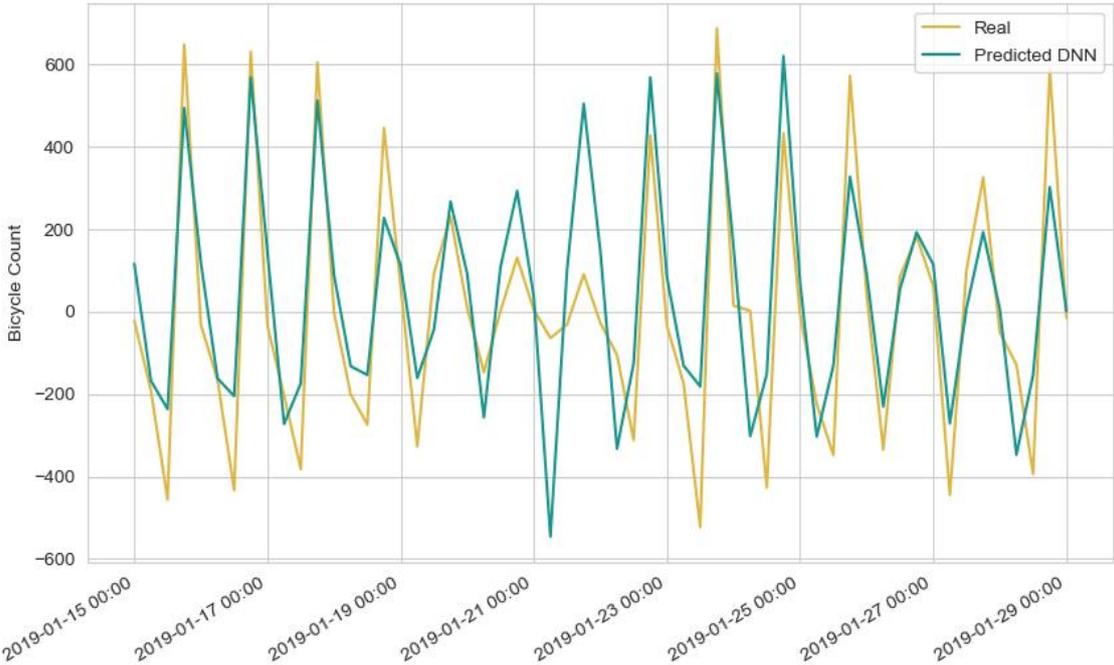


Fig. 4.12 City – level equilibrium prediction from 15/01/2019 to 29/01/2019 using Neural Networks.

4.1.3 Cluster – level prediction

The model has been slightly modified to carry out a cluster-based prediction. Each cluster is trained separately and divided in its sub-cluster. Thus, since in the case of New York 5 major clusters were selected (Fig. 4.13), we will have that same number of models. Each model will be able to predict the aggregated demand of each one of the sub-clusters included in the model. The minimum time-step considered in this case is 30min. The results are showcased in Table 4.6.

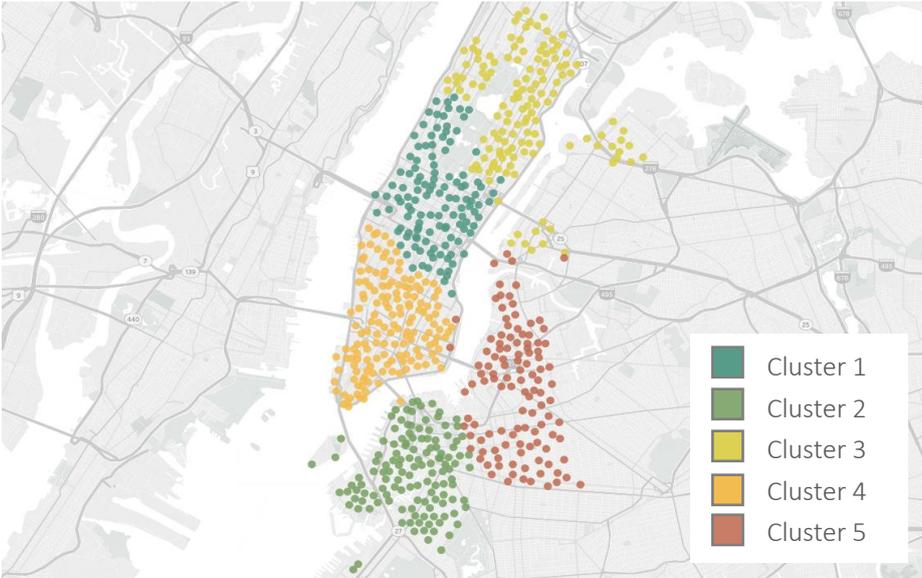


Fig. 4.13 Clusters used for New York City.

Table 4.6 Overall precision results for a cluster-level prediction of New York City.

Cluster	1	2	3	4	5	
N. Stations	123	129	135	153	114	
RF	Avg. Bikes	43,291	13,018	12,874	64,656	14,43
	Avg. Error	13,328	4,526	4,669	18,963	4,866
	Acc. (%)	69,21	65,24	63,73	70,67	66,28
	Max. Error	292,036	59,03	92,226	333,349	77,208
GB	Avg. Bikes	43,291	13,018	12,874	64,656	14,43
	Avg. Error	15,617	4,897	5,378	20,56	5,297
	Acc. (%)	63,92	62,38	58,24	68,2	63,29
	Max. Error	246,924	68,436	90,414	279,789	86,335
NN	Avg. Bikes	43,291	13,018	12,874	64,656	14,43
	Avg. Error	13,11	4,47	4,517	18,497	4,83
	Acc. (%)	69,72	65,67	64,92	71,39	66,53
	Max. Error	252,525	59,202	80,33	287,332	75,297

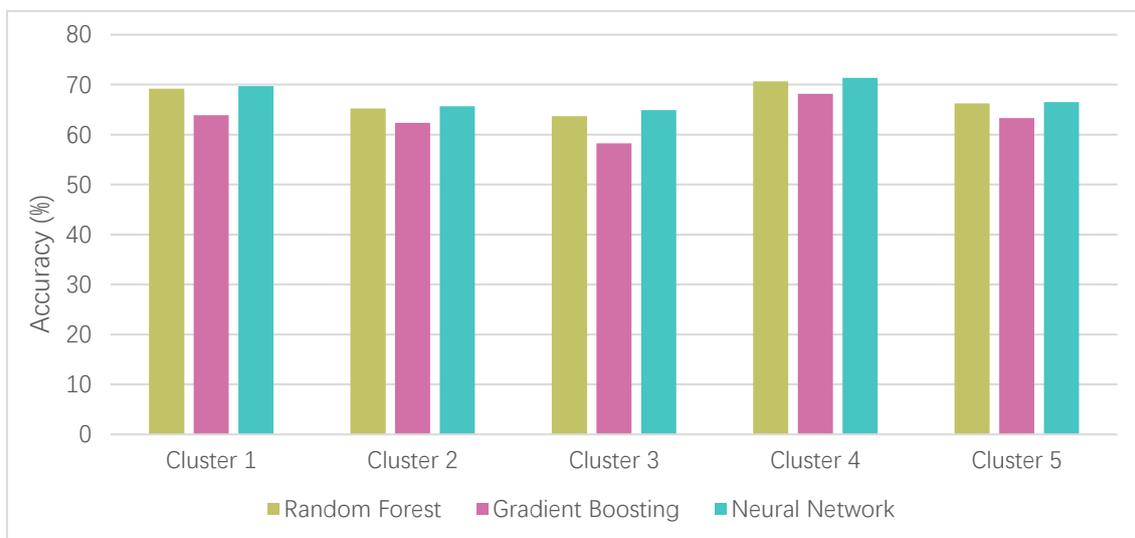


Fig. 4.14 Accuracy results for a cluster-level prediction of New York City.

From the obtained results, it can be observed that the Neural Network model yields the best results for each of the clusters without any exception, followed by the Random Forest model (Fig. 4.43). There are clearly two groups of clusters, ones with higher demand and others with lower one. Just checking the Avg. Bikes value, which corresponds to the average real number of bicycles being used, cluster 1 and 4 have almost five times the demand of clusters 2, 3 and 5, with the first ones corresponding to Midtown Manhattan. Since there is much more demand in the HD clusters, the average error is also higher, as well as the max error and the other error values.

Besides that, the accuracy is generally higher, with the HD clusters reaching around 70% precision, whereas the LD ones reach a maximum of 66%.

Since NN was able to reach the highest accuracy in all cases, the predictions showcased will only focus on this particular algorithm. The total amount of sub-clusters is 27, so only a selection of these will be analyzed into their full extent, selecting an example of a HD cluster and another example of a LD cluster. The dates predicted range from 15th January to 29th January (Fig. 4.44, Fig. 4.45)

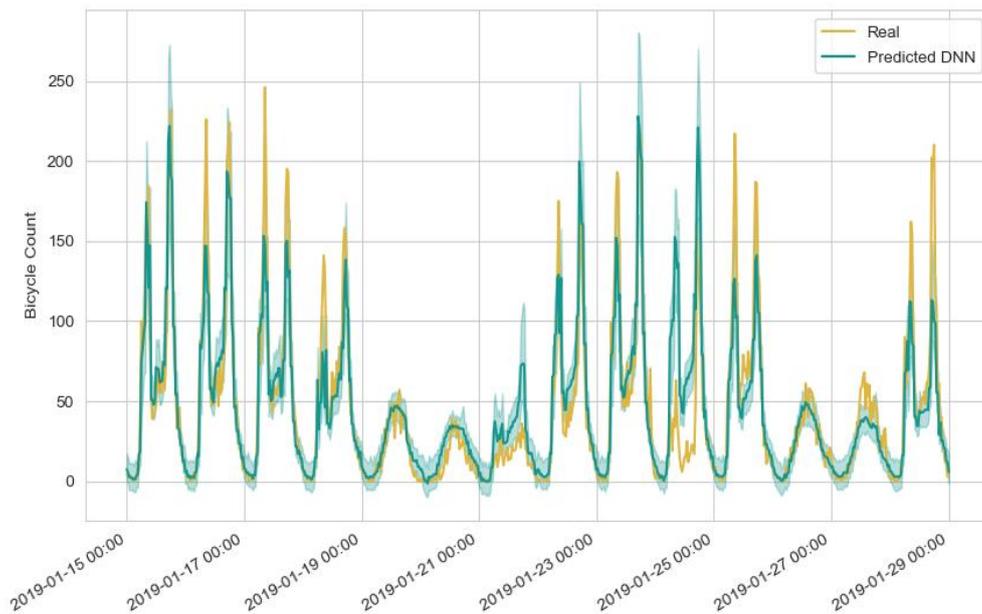


Fig. 4.15 HD Cluster – level demand prediction from 15/01/2019 to 29/01/2019 using Neural Networks.

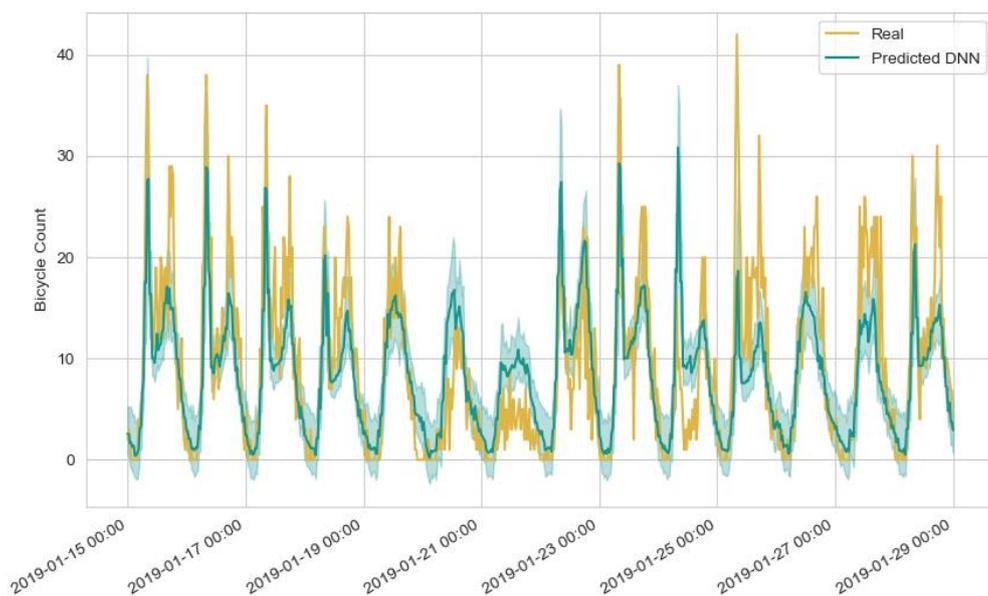


Fig. 4.16 LD Cluster – level demand prediction from 15/01/2019 to 29/01/2019 using Neural Networks.

Table 4.7 Cluster– level error results from 15/01/2019 to 29/01/2019 with a 30min time-step.

Cluster	Avg. Bikes	Avg. Error	Acc. (%)	Max. Error	RMSLE	RMSE
1.2 (HD)	44,676	15,078	66,25	137,348	0,210	0,487
3.1 (LD)	4,052	9,041	56,91	21,287	0,192	0,324

From Table 4.7 we can see that the accuracy in HD sub-clusters is higher than LD (around 10%), with the prediction being overall worse when comparing LD to HD. This is regardless the amount of stations each cluster has, with the 1.2 sub-cluster being made of the sum of the demand of 18 stations, whereas the 3.1 cluster is made up of 36 stations. Thus, having an overall lower demand makes it difficult for the algorithm to reach the same level of accuracy that usually a HD sub-cluster would yield, since each station has more variability in their patterns. In particular, for the LD sub-cluster to reach a closer accuracy level to the HD one, a time-step of 1h yields an overall accuracy of 61,74, which is still lower than the original. This means that, when sub-aggregating even more (to station level), there will be a considerable difference between time-steps when considering LD stations and HD ones.

4.1.4 Station – level prediction

The minimum time-step chosen for a station-level prediction is at least one hour. The showcased predictions correspond to a station from the sub-clusters analyzed in part 4.1.4, one for the HD cluster (1.2) and LD one (3.1). First, the error results for the three tested algorithms are showcased for the validation test (Table 4.8).

Based on the test set, Gradient Boosting obtains the worst results for both cases, whereas Random Forest and Neural Network almost obtain the same results, the second one yielding slightly less error. It should be noted that the time-step is changed from 1h in the HD station compared to the 3h in the LD, which still showcases less accuracy. If a smaller time-step is chosen for the LD station, even aggregating the prediction does not yield to higher accuracy, since the initial prediction did not reach a minimum accuracy threshold. This depends on the total number of bikes being used in that time-step. In comparison, the HD cluster analyzed has 7,70 average bikes in one hour, whereas the LD one only has 4,46 bikes in three hours, which corresponds to almost five times less information when compared to the first one.

Table 4.8 Station– level error results with a 1h time-step (HD) and 3h time-step (LD).

	Cluster	Avg. Bikes	Avg. Error	Acc. (%)	Max. Error	RMSLE	RMSE
Random Forest	1.2 (HD)	7,699	3,337	56,65	191,104	0,223	0,565
	3.1 (LD)	4,457	2,117	52,51	35,539	0,247	0,569
Gradient Boosting	1.2 (HD)	7,699	3,771	51,02	193,133	0,238	0,596
	3.1 (LD)	4,457	2,218	50,22	33,728	0,249	0,577
Neural Network	1.2 (HD)	7,699	3,320	56,87	185,601	0,220	0,546
	3.1 (LD)	4,457	2,116	52,51	33,51	0,245	0,560

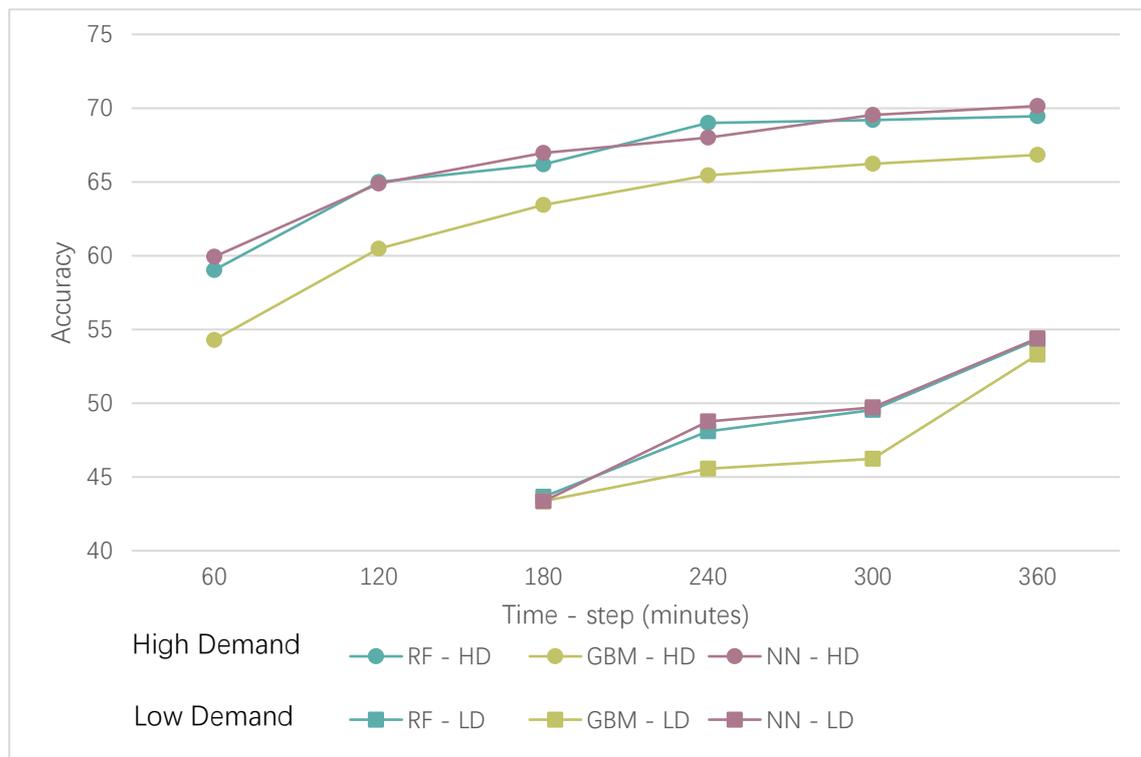


Fig. 4.17 Accuracy evolution when increasing the time-step interval for HD and LD stations.

In particular, for a prediction of the last two weeks of January (15th to 29th of January), we can obtain the error results for station 402 from the HD cluster (0) and for the station 3120 from the LD cluster. For the HD station, 1h time-step is chosen as the minimum one, but as for the LD one this is not enough (Fig. 4.46), so a 3h time-step is chosen instead.

From these results, we can obtain some kind of idea of what time-step to use to reach a convergent accuracy when further aggregating predictions, which will be needed when predicting the demand in each station. Obviously, the average number of bikes is not constant in a year, thus a same station can have a different minimum time-step when predicting different time-intervals. For this reason, it is recommendable to first obtain the minimum average number of bikes for a station in one hour, which would correspond to a time of the year with low demand.

Table 4.9 Relationship between the average number of bikes in a and minimum time-step required.

Average Number of bikes (in 1h)	Minimum time – step (h)
$X \geq 7$	1
$1,5 \leq X < 7$	2
$X < 1,5$	3 or more

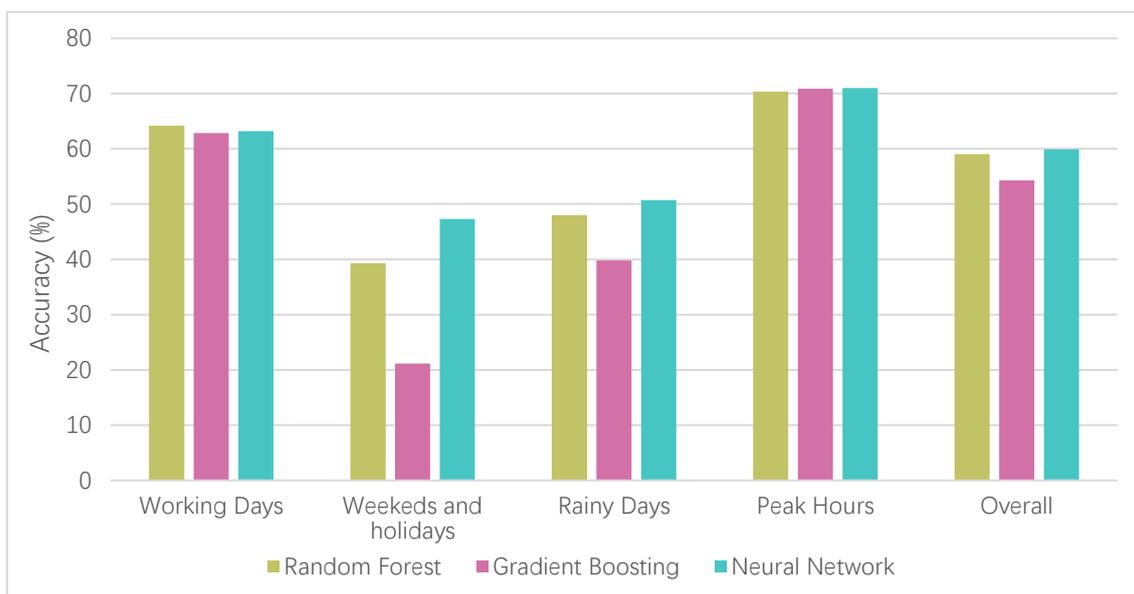


Fig. 4.18 Station 402 average accuracy results from 15/01/2019 to 29/01/2019 with a 1h time-step.

Table 4.10 Station 402 error results from 15/01/2019 to 29/01/2019 with a 1h time-step.

		Random Forest		Gradient Boosting		Neural Network	
Demand		In	Out	In	Out	In	Out
Working Days	Avg. Bikes	9,309	8,922	9,309	8,922	9,309	8,922
	Avg. Error	3,649	2,899	3,516	3,255	3,551	3,164
	Acc. (%)	60,8	67,51	62,23	63,51	61,85	64,54
	Max. Error	38,204	18,744	37,692	21,641	38,227	21,94
Weekends and	Avg. Bikes	4,333	4,217	4,333	4,217	4,333	4,217
	Avg. Error	2,661	2,534	3,678	3,072	2,163	2,337
	Acc. (%)	38,6	39,91	15,13	27,15	50,09	44,59
	Max. Error	29,374	24,658	31,263	29,004	13,088	16,977
Rainy days	Avg. Bikes	5,679	5,732	5,679	5,732	5,679	5,732
	Avg. Error	3,292	2,638	3,728	3,137	2,957	2,662
	Acc. (%)	42,03	53,97	34,34	45,27	47,93	53,56
	Max. Error	38,204	24,658	37,692	29,004	38,227	16,977
Peak Hours	Avg. Bikes	20,222	27,667	20,222	27,667	20,222	27,667
	Avg. Error	7,485	6,171	6,842	6,737	6,582	7,026
	Acc. (%)	62,98	77,7	66,17	75,65	67,45	74,6
	Max. Error	25,191	18,744	23,523	20,221	26,634	20,082
Overall	Avg. Bikes	7,537	7,246	7,537	7,246	7,537	7,246
	Avg. Error	3,297	2,769	3,574	3,19	3,057	2,869
	Acc. (%)	56,26	61,79	52,59	55,98	59,44	60,4
	Max. Error	38,204	24,658	37,692	29,004	38,227	21,94

Firstly, the results of station 402 (HD) are presented in 0 for both demands getting into the station and out of it. The average accuracy is represented in Fig. 4.48 as reference and easy comparison. As expected, Gradient Boosting got the overall worst results, only reaching an overall accuracy of 54,285, compared to the 59,025 obtained with the Random Forest and 59,920% with the Neural Network, which yielded the best results.

If we disaggregate the analyzed intervals, the highest accuracy is obtained for the peak hours followed by the working days, whereas the worst one is obtained for weekends and holidays. This goes in like with the hypothesis that the higher the average number of bikes in the analyzed time-step, the higher the accuracy. But this is not the only variable that should be taken into account, since, overall, demand out of station yields higher accuracy than the demand in the station, even

though the average number of bikes out of the station is lower than the bikes in ($7,246 < 7,537$). This might be explained by the fact that the demand out of the station is extremely concentrated during peak hours, which has a higher number of average bikes out of the station rather than into it ($27,667 > 20,222$).

Further examples of the results for station 402 can be seen in Fig. 4.19, Fig. 4.20 and Fig. 4.21. As it is showcased, the algorithms mostly predict a “regularized” version of the demand curve, that is without the ups and downs between single hours. Gradient Boosting seems to predict the best the peak hours (as already showcased in 0), whereas RF has the highest error in these particular hours. Overall, the Neural Network layout is able to get the highest accuracy.

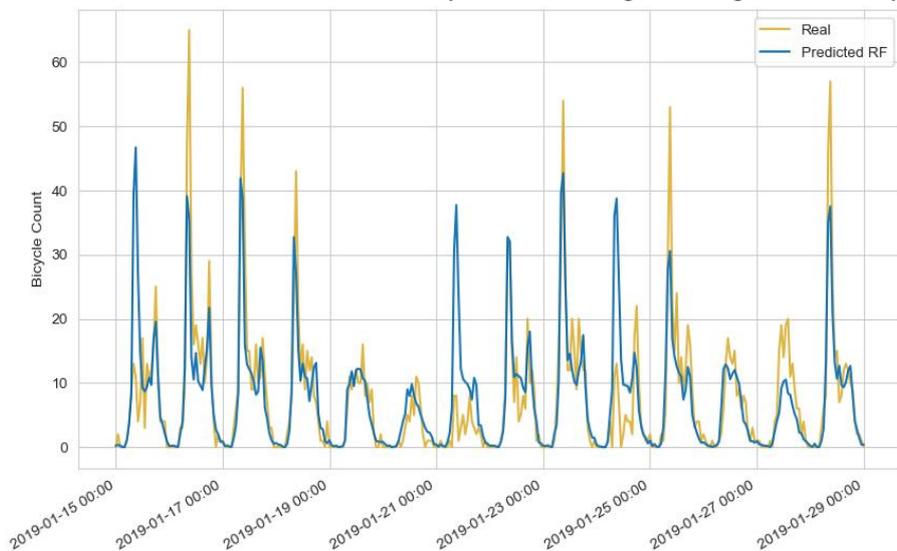


Fig. 4.19 Station 402 (HD) demand (in) prediction from 15/01/2019 to 29/01/2019 using RF.

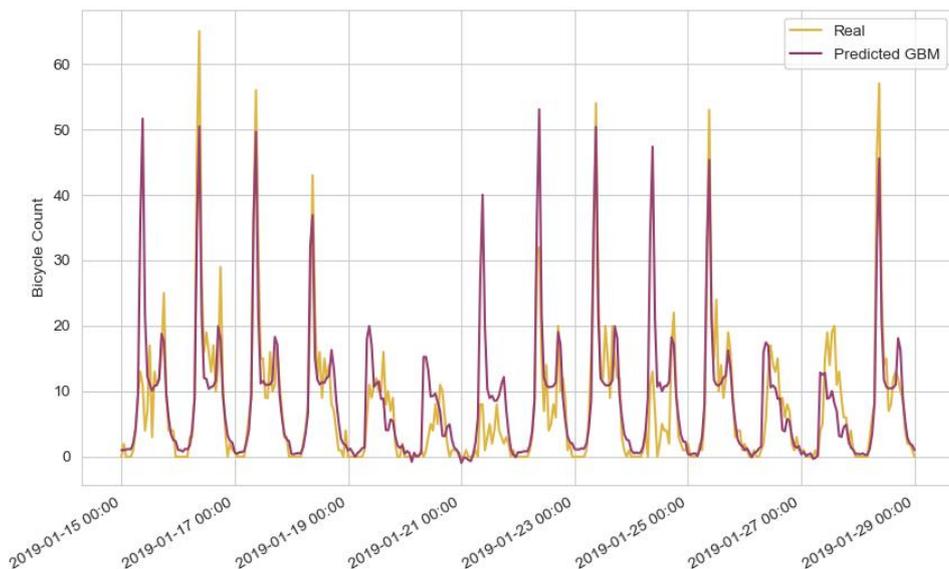


Fig. 4.20 Station 402 (HD) demand (in) prediction from 15/01/2019 to 29/01/2019 using GBM.

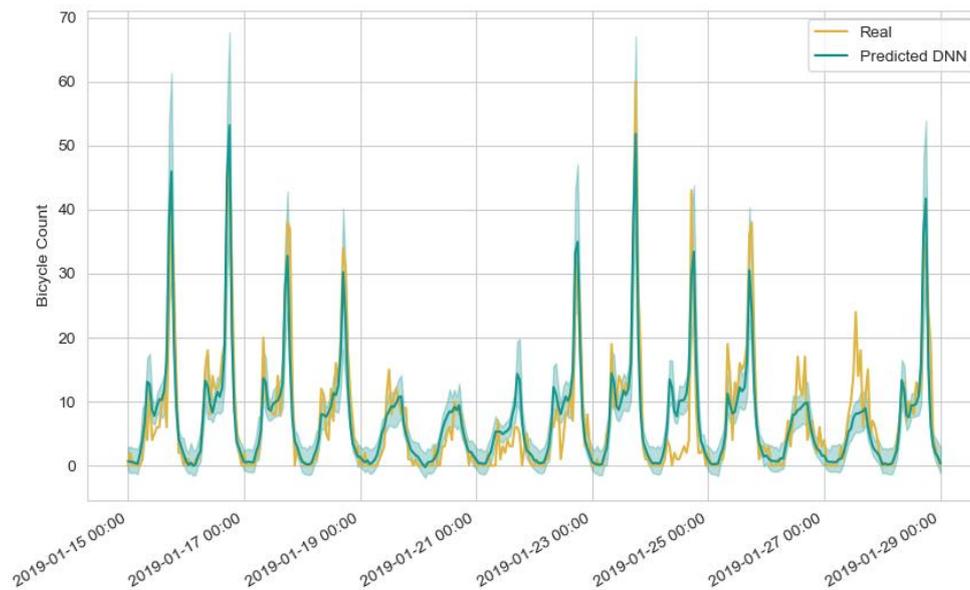


Fig. 4.21 Station 402 (HD) demand (out) prediction from 15/01/2019 to 29/01/2019 using NN.

Similarly, we can obtain the same results for station 3120, corresponding to the LD cluster (Table 4.11, Fig. 4.22). Again, some examples are showcased in Fig. 4.23, Fig. 4.24 and Fig. 4.25.

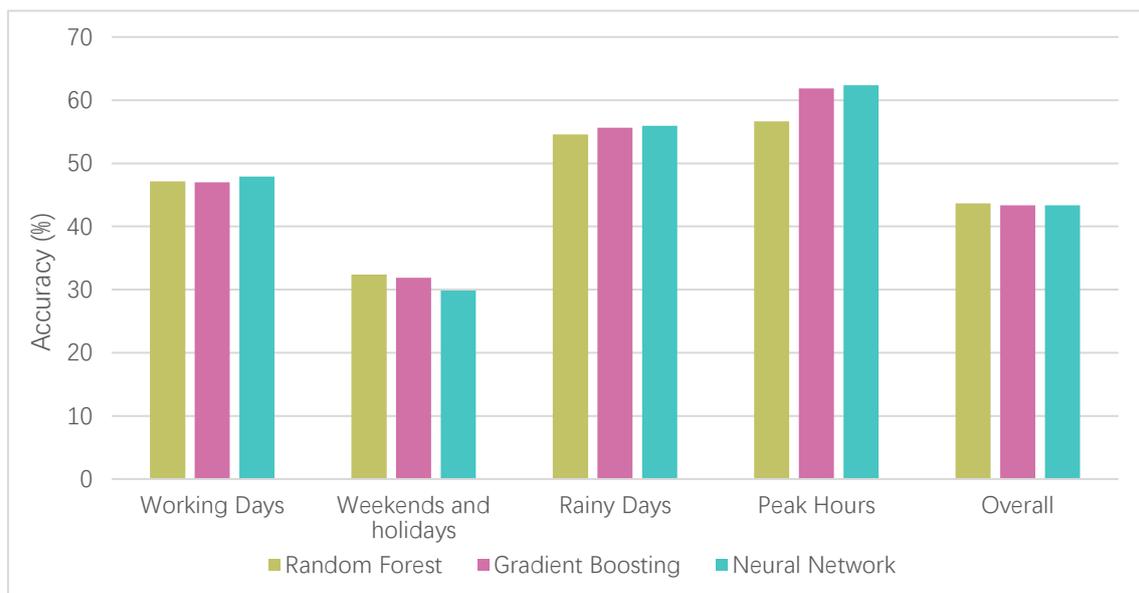


Fig. 4.22 Station 3120 average accuracy results from 15/01/2019 to 29/01/2019 with a 3h time-step.

From the accuracy results regarding the LD station, we can see that the overall precision has descended almost a 20% just by the fact that there is less data available in this type of stations, and even considering that the time step is now 3h instead of 1h. In particular, the algorithms have a similar overall behavior, with NN excelling in peak hour prediction and working days, whereas RF has better results on weekends and holidays and GB in rainy days.

Table 4.11 Station 3120 error results from 15/01/2019 to 29/01/2019 with a 3h time-step.

		Random Forest		Gradient Boosting		Neural Network	
Demand		In	Out	In	Out	In	Out
Working Days	Avg. Bikes	3,413	3,115	3,413	3,115	3,413	3,115
	Avg. Error	1,764	1,682	1,805	1,657	1,764	1,635
	Acc. (%)	48,32	46,0	47,13	46,82	48,33	47,53
	Max. Error	11,136	8,484	9,733	6,877	10,056	6,506
Weekends and	Avg. Bikes	2,579	2,965	2,579	2,965	2,579	2,965
	Avg. Error	2,072	1,626	2,02	1,718	2,094	1,752
	Acc. (%)	19,67	45,15	21,67	42,05	18,81	40,92
	Max. Error	7,368	6,396	8,03	5,571	6,824	5,931
Rainy days	Avg. Bikes	3,365	3,104	3,365	3,104	3,365	3,104
	Avg. Error	1,575	1,368	1,474	1,393	1,485	1,365
	Acc. (%)	53,19	55,94	56,21	55,12	55,85	56,03
	RMSE	0,418	0,35	0,37	0,333	0,384	0,326
Peak Hours	Avg. Bikes	7,071	4,929	7,071	4,929	7,071	4,929
	Avg. Error	3,158	2,071	2,832	1,784	2,807	1,752
	Acc. (%)	55,34	57,97	59,95	63,8	60,31	64,46
	Max. Error	11,136	8,484	10,056	6,506	9,733	6,464
Overall	Avg. Bikes	3,112	3,053	3,112	3,053	3,112	3,053
	Avg. Error	1,848	1,628	1,839	1,655	1,836	1,658
	Acc. (%)	40,62	46,68	40,93	45,79	41,02	45,69
	Max. Error	11,136	8,484	9,733	6,877	10,056	6,506

From the results table (Table 4.11) we can further see that this particular station is quite in equilibrium, since the number of average bikes (the real one) is similar for trips into the station as well as outside. The only noticeable difference is that, during peak hours, there are more trips into the station rather than out of it, which might be possible if there is some type of working place or office nearby that attracts this kind of demand in those time intervals.

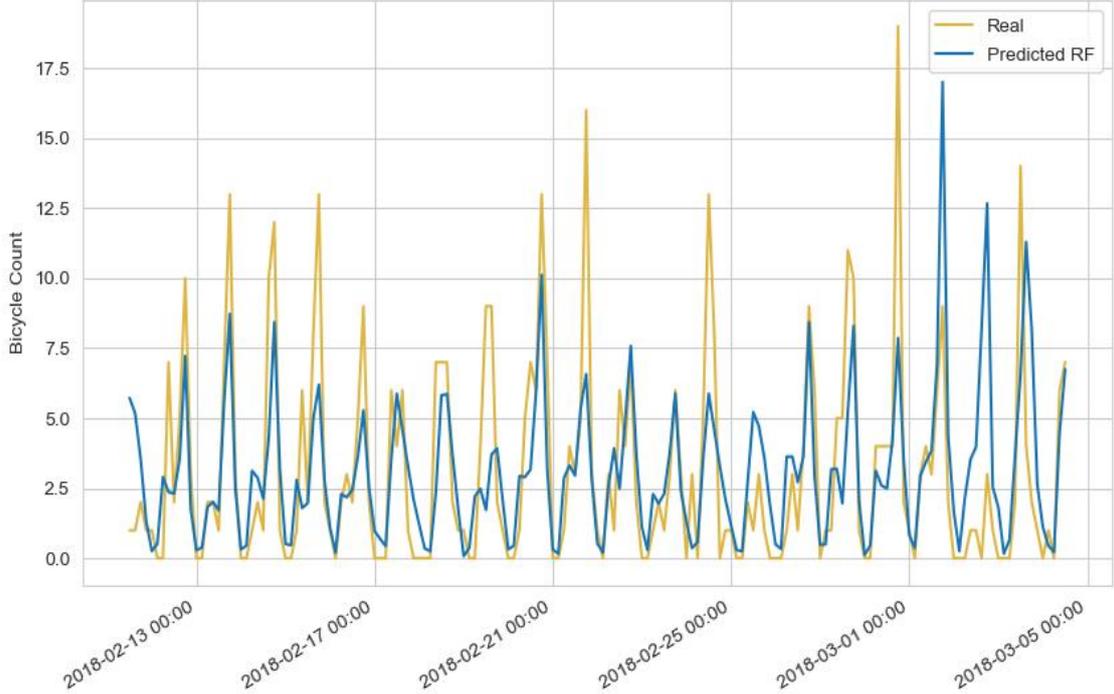


Fig. 4.23 Station 3120 (LD) demand (in) prediction from 15/01/2019 to 29/01/2019 using RF.

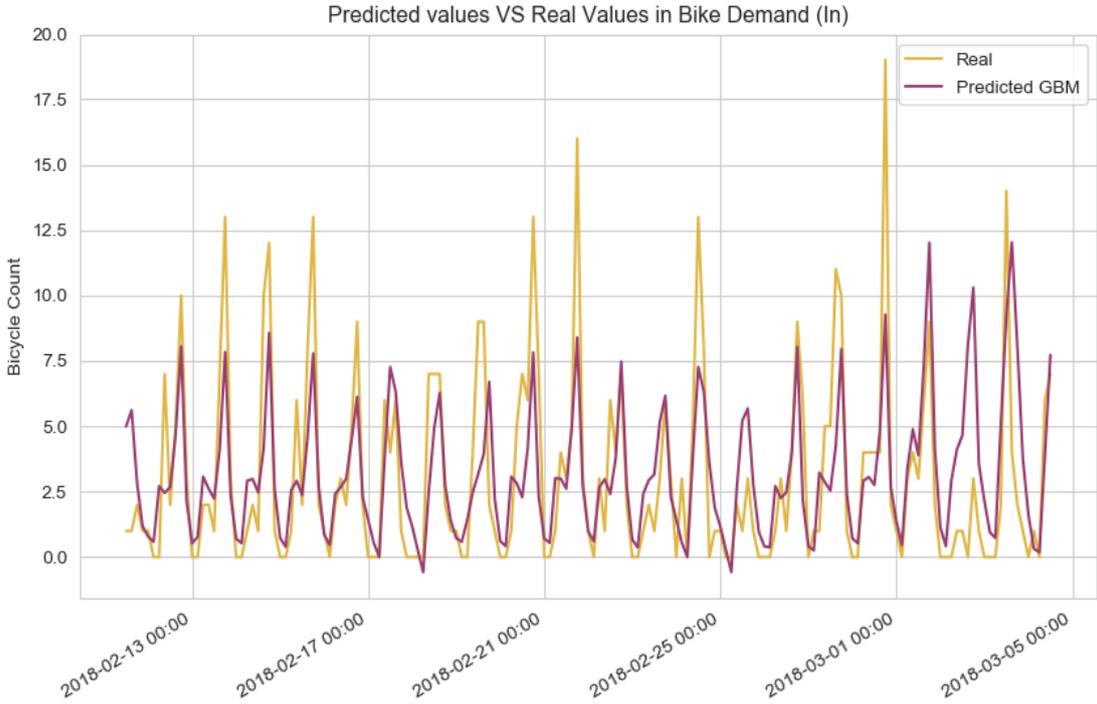


Fig. 4.24 Station 3120 (LD) demand (in) prediction from 15/01/2019 to 29/01/2019 using GBM.

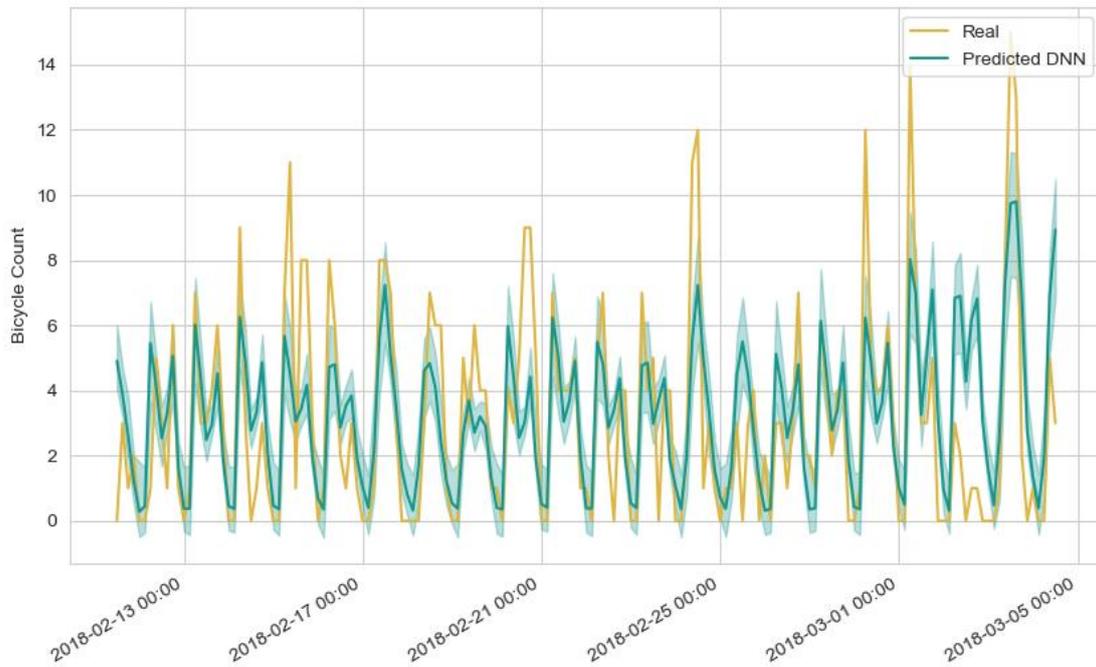


Fig. 4.25 Station 3120 (LD) demand (out) prediction from 15/01/2019 to 29/01/2019 using NN.

One can also calculate the difference between the demand into the station and out of the station (clean demand), to try to predict the extra number of bikes in a station at a certain time interval. As it was mentioned before, this operation usually yields a lower accuracy than the obtained in the prediction of the demand itself. Thus, for this part, a time-step of 24h is chosen, to simulate the status of a station during midnight, possible time for relocation to happen.

The results in Fig. 4.26 showcase that the predicted values tend to underestimate the real ones, which match to the predictions obtained until now, where it was difficult for the algorithm to reach the maximum peaks of demand. This underestimating error keeps adding up through the day, which yields to the overall underestimation of the clean demand in the city. The error is more noticeable in the city center, where the predicted demand volumes are higher (Fig. 4.27).

It should be noted that the predicted values have been rounded to the nearest integer for proper comparison. It could be possible that applying a corrective factor before rounding, which yields with the underestimation, would yield better results.

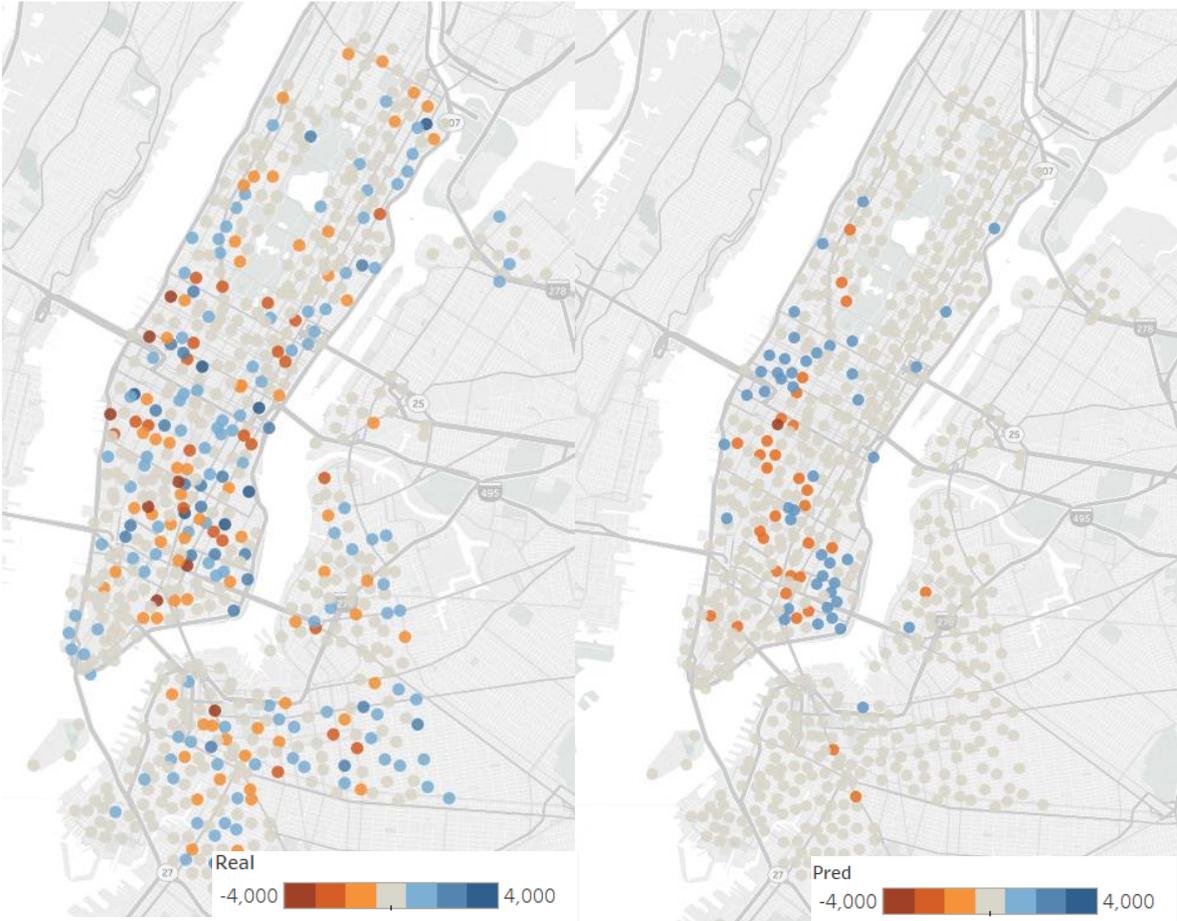


Fig. 4.26 Clean real (left) and predicted (right) demand for 15th of January.

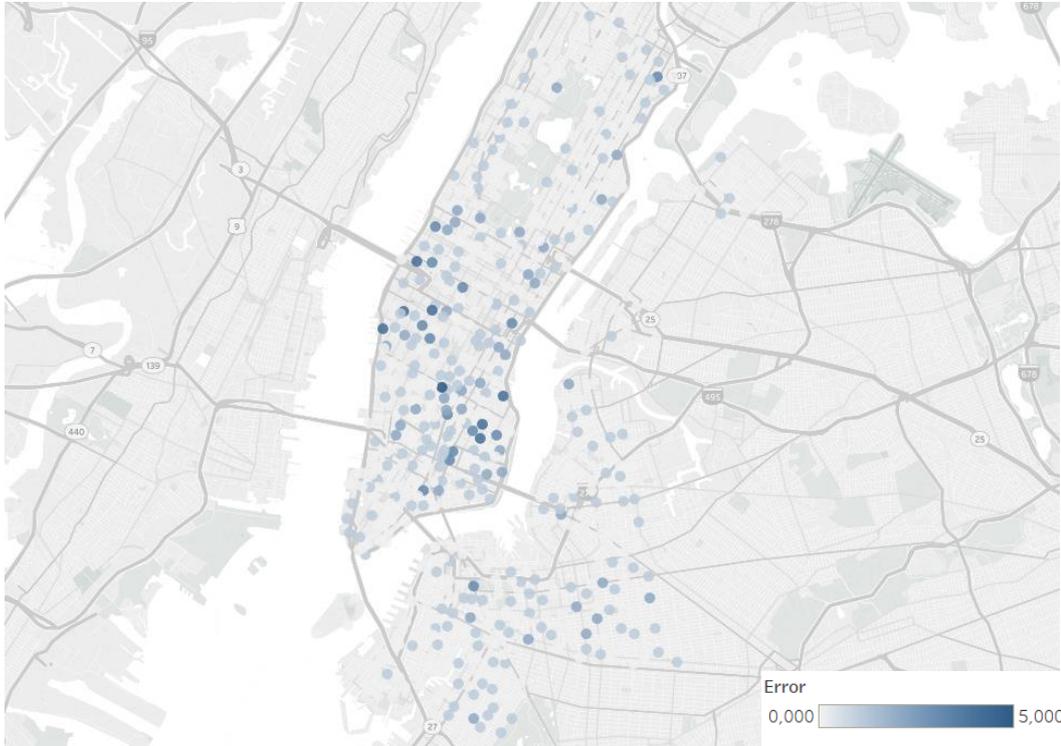


Fig. 4.27 Clean demand error in New York City for 15th of January.

Finally, the weather data used for all these predictions thus far has been based on historical data, rather than predicted one. This is important to consider, since the predicted information will inherently have error which could affect the prediction accuracy. Even though by discretizing the weather variables part of this error is ignored, the same period that has been analyzed during this study case (second week of January), will be predicted using weather predictions instead (Table 4.12). The predictions are carried out using the NN algorithm, since it was the one that presented the highest accuracy, and for the HD station analyzed (402).

Table 4.12 Prediction results for 15/01/2019 to 21/01/2019 with different weather percentage error.

	Error	0%	25%	50%	75%	100%
Temperature	Avg. Bikes	7,036	7,036	7,036	7,036	7,036
	Avg. Error	3,082	2,716	2,949	3,163	2,958
	Acc. (%)	56,2	61,39	58,09	55,04	57,96
	Max. Error	27,977	27,575	25,648	21,384	29,877
	RMSLE	0,231	0,206	0,226	0,226	0,235
	RMSE	0,477	0,389	0,451	0,434	0,456
Wind Speed	Avg. Bikes	7,036	7,036	7,036	7,036	7,036
	Avg. Error	3,082	3,058	3,072	3,068	2,968
	Acc. (%)	56,2	56,53	56,34	56,39	57,82
	Max. Error	27,977	22,453	27,732	19,688	23,211
	RMSLE	0,231	0,22	0,227	0,221	0,217
	RMSE	0,477	0,441	0,463	0,428	0,446

From the obtained results, it is clear that the algorithms are extremely robust against errors in the weather forecast. The small changes in accuracy are probably more due to rebuilding of the model (every time it is rebuilt there are small changes in the outcome) rather than the inaccuracy of the weather forecast.

This robustness was proved only for the wind and temperature variables, rather than the rain or snow amount. This is due to these two other variables having a great impact on the overall demand prediction. Thus, the model will be robust up until the point when precipitation is predicted feasibly. Otherwise, the accuracy will drop for those time-steps when the precipitation forecast is wrong.

4.2 London

London's own bike sharing system started operations in July 2019 under the name of Barclays Cycle Hire, with a total of 5000 bicycles and 315 available docks. It has received several expansions and now it counts with more than 11000 bikes and 800 stations, becoming the largest cycle hire scheme in Europe [44].

As other systems, there are two type of users, differentiated between usual customers and single-time riders. The later can just pay £2 to access the bikes for up to 24h, with a flat fee during the first 30min and a surcharge of £2 for every 30min. On the other hand, regular members can pay a yearly fee of £90 to get unlimited use of the bike system, with a maximum trip. This fee can result more economic if you are student, since you can claim a 25% discount.

4.2.1 Data exploring

4.2.1.1 Weather

As done in the previous case, the weather conditions are discretized based on their quantile distribution. Results are presented in Table 4.13.

Table 4.13 Weather conditions discretization.

Category	Temperature (°C)		Wind (m/s)		Rain (mm)		Snow (cm)	
	L _{Boundary}	U _{Boundary}						
0	-	5,62	-	2,2	-	0,001	-	0,001
1	5,62	9,055	2,2	4,4	0,001	0,0229	0,001	50
2	9,055	12,76	4,4	6,6	0,0299	0,0838	-	-
3	12,76	17,776	6,6	8,8	0,0838	0,239	-	-
4	17,776	-	8,8	-	0,239	-	-	-

4.2.1.2 Demand behavior

Before carrying out any kind of prediction, it is necessary to understand how the system behaves in order to be able to reach accurate conclusions. This is done by calculating the number of bikes being used in the city depending on different variables, such as day of the week or season.

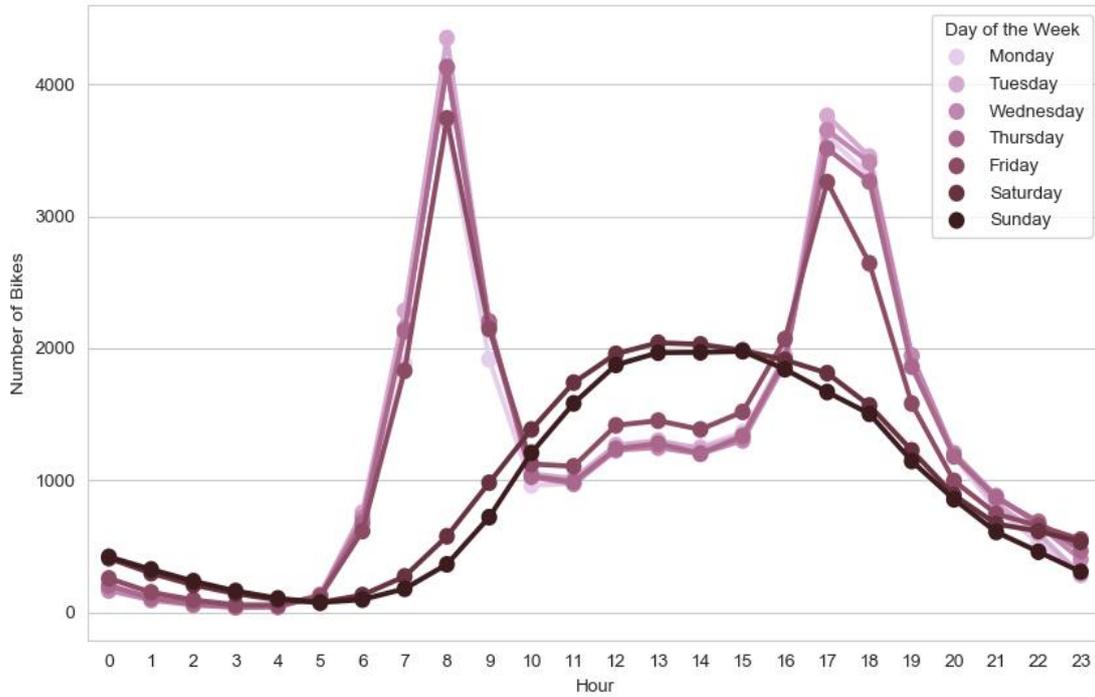


Fig. 4.28 Distribution of the bikes in use depending on the weekday.

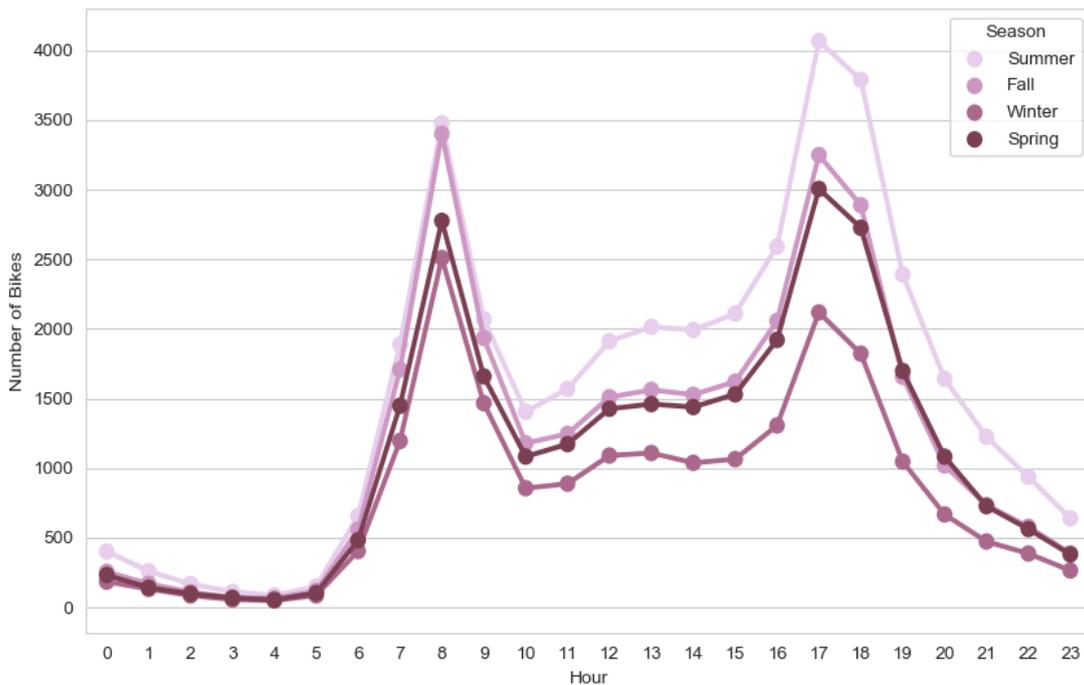


Fig. 4.29 Distribution of the bikes in use depending on the season.

From Fig. 4.28 and Fig. 4.29, we can see that the temporal distribution quite resembles the one in New York, with peak hours around 8am and 5 – 6pm during weekdays and a smoother distribution during weekends. As for the seasons, summer is peak season and winter showcases the less demand, thus there is an important weather factor to take into account.

4.2.1.3 Spatial behavior

Plotting the total number of generated trips as well as the received trips for each station, we can check the general equilibrium of the system, as well as the busiest stations (Fig. 4.30).

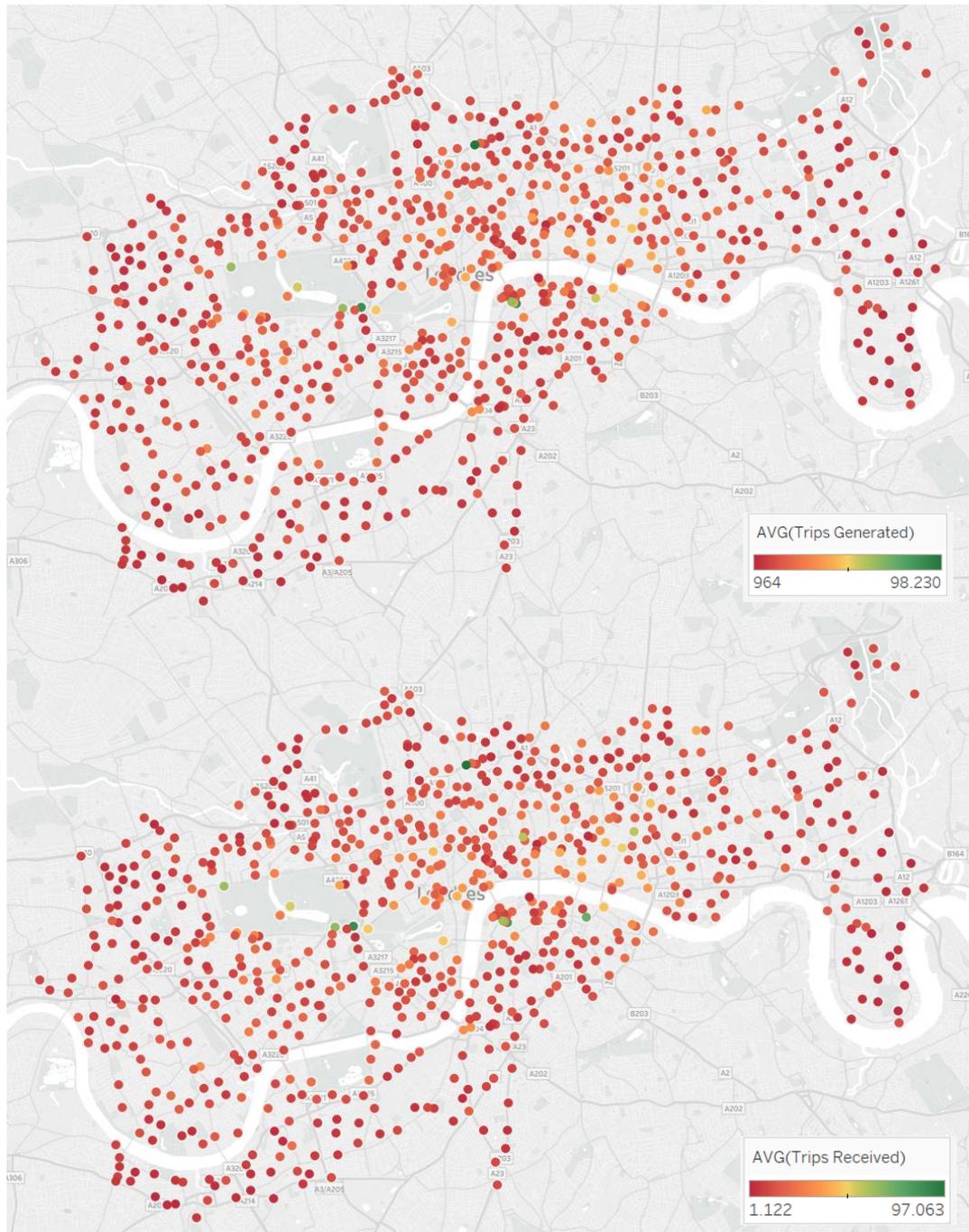


Fig. 4.30 Trips received (top) and trips generated (bottom) in London during a year.

From the image, we can observe that the majority of the system has a quite uniform demand, but several stations around the center exhibit a high number of generated and received trips, with a difference of almost of 100 thousand trips a year between the less used to the most popular

station. This more popular stations will exhibit more reliable demand patterns, thus smaller time steps might be considered, whereas for the stations in the outskirts larger time-steps should be considered. It should be noted that the stations that exhibit the lower demand might because they were not open for the total of the analyzed time period, which makes them appear less popular when comparing the total amount of demand.

If plotting the demand equilibrium (generated demand minus generated demand) we can further explore the operation conditions of the system (Fig. 4.31). From there we appreciate that center of London has a center of demand generated with several stations exhibiting a quite high disequilibrium. As for the receiving demand centers, they seem to be situated right outside the generating center and on the north part of the city.

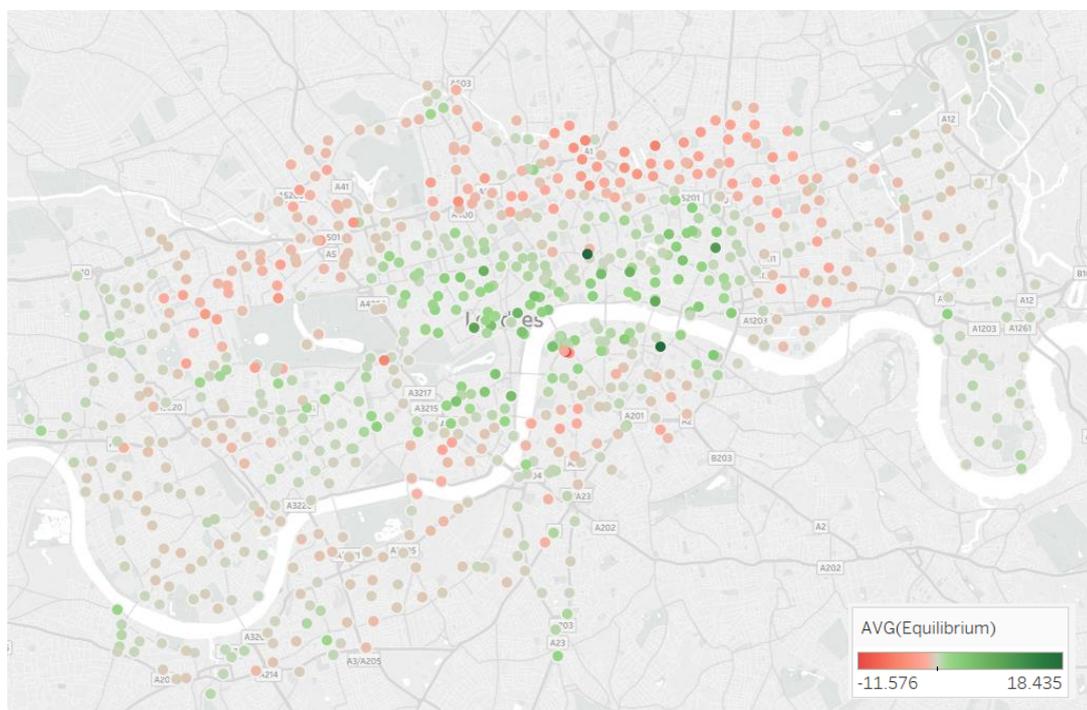


Fig. 4.31 System equilibrium (trips received minus trips generated).

Finally, the capacity of the stations and average trip duration is also plot to further test some hypothesis, such as the highest capacity stations are in the zones with higher demand and that the longest rides usually take place around green zones or stations further away from the center.

In the first picture (Fig. 4.32), we can observe that stations around the center may exhibit higher capacity, even though some stations around the outskirts of the city also do, which might be due to the lack of station density in the area, which implies that the present stations should have a higher capacity.

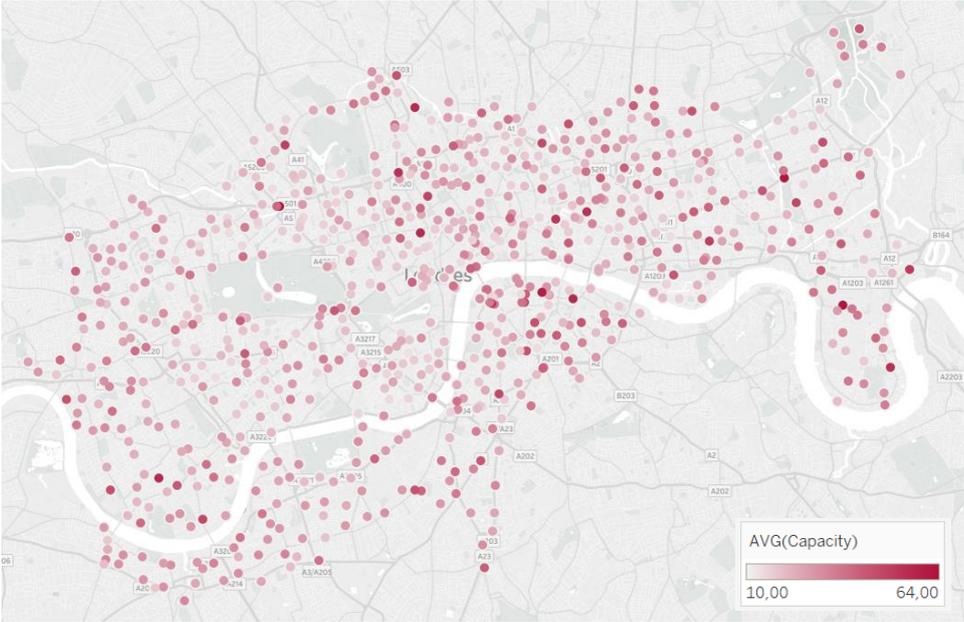


Fig. 4.32 Capacity of London’s bike stations.

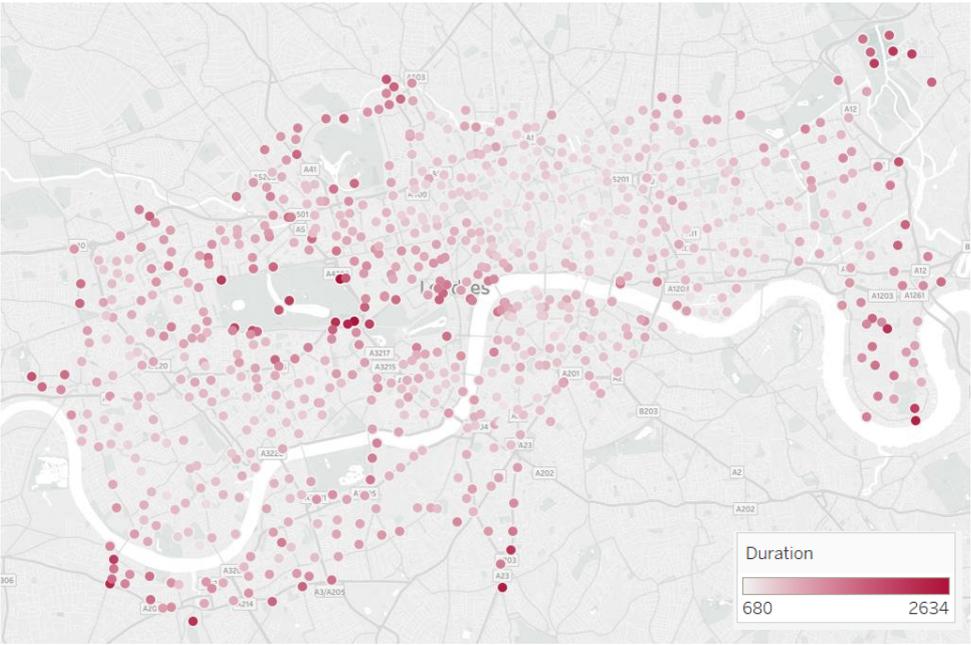


Fig. 4.33 Average duration of rides in London (seconds).

In the second picture (Fig. 4.33), we can appreciate the longest trips are carried out around Hyde Park, which might be a more recreational use, and the most outskirts parts of the city, since presumably citizens could easily cycle from their homes to the center. This particular behavior is different than in New York City, where the distance from the outskirts to the center are too large to regularly bike and there is a cap on the maximum use time of 45min.

4.2.2 City – level prediction

The city – level prediction is initially done with a time-step of 15min. The results are presented for the validation set, again corresponding to the 20% of the total set. Since both demand into the system and out of it are calculated, the overall error results are presented as the average between both.

Table 4.14 Overall precision results for a city-level prediction of London.

	Time (s)	Acc. (%)	Avg. Error	Expl. Var	Max. Error	RMSLE	RMSE
Random Forest	20,52	80,58	57,192	0,886	823,274	0,159	0,337
Gradient Boosting	26,64	77,79	65,409	0,878	731,061	0,164	0,348
Neural Network	49,716	83,12	49,716	0,927	719,236	0,138	0,268

* Average number of bikes equal to 294,50 bikes.

From Table 4.14, we can see that the best accuracy is reached with a Neural Network (83,12%) and Random Forest comes second with an 80,58% of accuracy. In general, NN performs better in every aspect, even though it takes double the time to build the model. Compared to the case of NY, the maximum reached accuracy is 3% higher, taking into account that the average number of bikes in London is almost half of the case in New York.

Again, the last two weeks of January are chosen to further compare with the results obtained in the New York case. The figures showcase the particular case of demand received, since both cases are too similar in a city-level prediction (Fig. 4.34, Fig. 4.35 and Fig. 4.36).

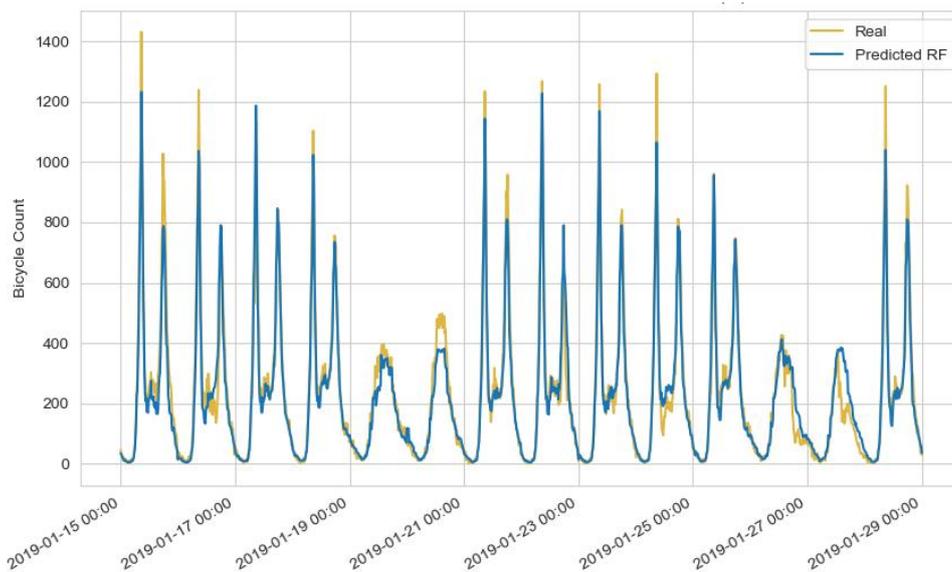


Fig. 4.34 City – level demand prediction from 15/01/2019 to 29/01/2019 using Random Forest.

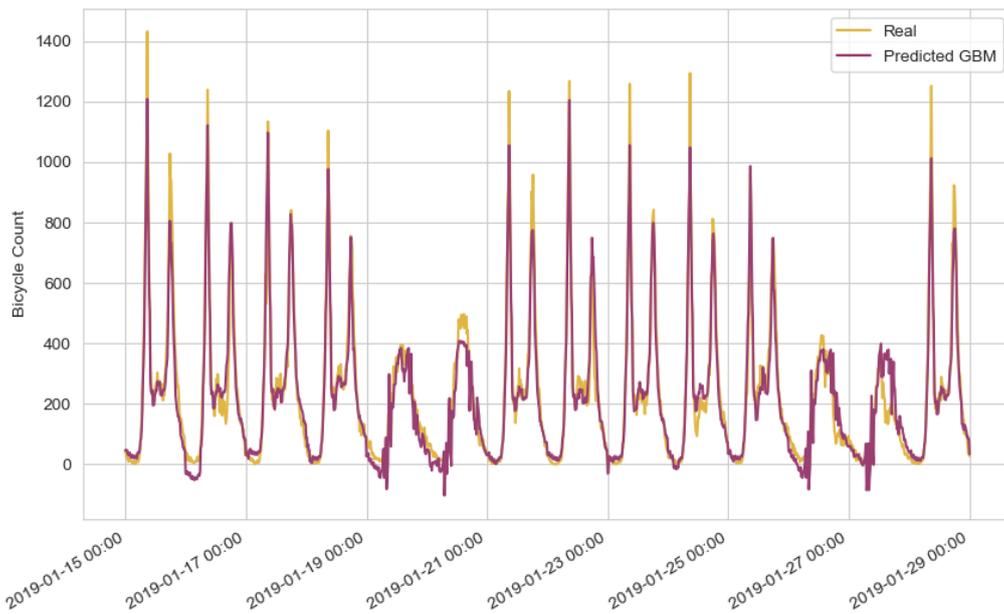


Fig. 4.35 City – level demand prediction from 15/01/2019 to 29/01/2019 using Gradient Boosting.

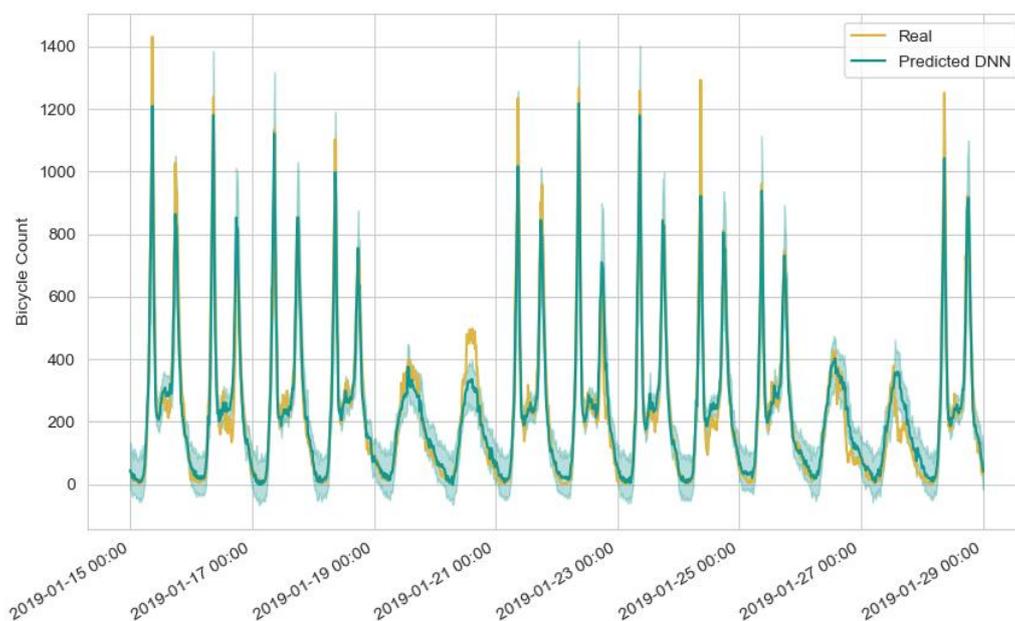


Fig. 4.36 City – level demand prediction from 15/01/2019 to 29/01/2019 using Neural Networks.

From the predictions, we can observe that they all barely look any different, with NN being the prediction that can reach the maximums better. Besides that, Gradient Boosting seems to predict some negative values instead of zeroes, but the overall prediction is acceptable.

To further compare the obtained results, the errors will be calculated on different time periods, such as working days or weekends. (Table 4.15). During this time period, it did not rain neither snow in London, thus that calculation is not carried out in this case.



Fig. 4.37 City – level accuracy results from 15/01/2019 to 29/01/2019.

Table 4.15 City – level error results from 15/01/2019 to 29/01/2019.

		Random Forest	Gradient Boosting	Neural Network
Working Days	Avg. Bikes	262,871	262,871	262,871
	Avg. Error	31,322	36,945	33,545
	Acc. (%)	88,08	85,95	87,24
	Max. Error	250,261	262,552	372,078
Weekends and	Avg. Bikes	152,586	152,586	152,586
	Avg. Error	34,051	48,643	37,779
	Acc. (%)	77,68	68,12	75,24
	Max. Error	240,495	240,944	214,504
Peak Hours	Avg. Bikes	756,6	756,6	756,6
	Avg. Error	75,78	83,086	79,662
	Acc. (%)	89,98	89,02	89,47
	Max. Error	250,261	262,552	372,078
Overall	Avg. Bikes	231,384	231,384	231,384
	Avg. Error	32,101	40,285	34,754
	Acc. (%)	86,13	82,59	84,98
	Max. Error	250,261	262,552	372,078

From the error results obtained from the prediction, one can see that Random Forest and gets the best results, with Neural Networks being the second (Fig. 4.37). This is true for every aspect analyzed and every time period analyzed, which is a different conclusion when analyzing the case of New York City, where depending on the time – period analyzed, some algorithms were better than the others.

To fully understand the effects of the chosen time-step, the same simulation carried out with a 15min time-step is aggregated in larger time-steps for the three algorithms analyzed (Fig. 4.38).

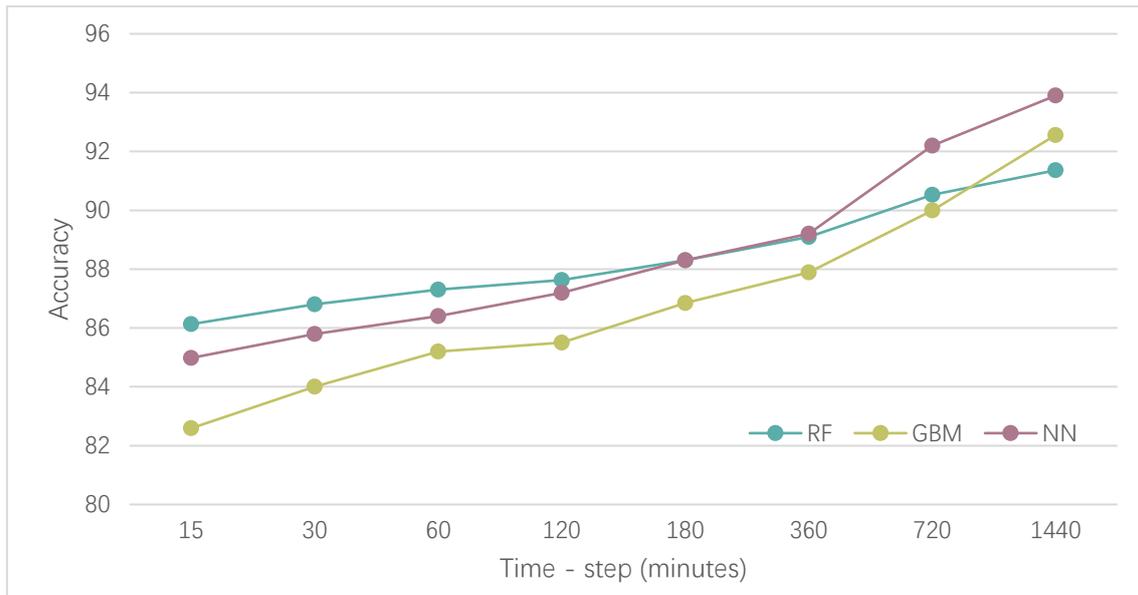


Fig. 4.38 Accuracy with increased time – step in the city-level prediction from 01/01/2019 to 15/01/2019.

As previously discussed, a larger time-step yields to a higher accuracy in either of the algorithms used to predict the real demand value, with NN exhibiting the highest accuracy for larger time-steps, and RF for smaller ones. In this particular case there is a higher level of accuracy compared to NY, which might be due to the lack of rain during the analyzed weeks, a phenomenon that really disrupts the prediction precision. Besides that, with an aggregation of 24h, the maximum accuracy level reached is of 93%, which is around 7% more than the case for 15min. This demonstrates that a small time-step in this type of predictions is good enough to reach high levels of accuracy.

Finally, for this city-level prediction one can try to predict the difference between the generated and received trips, thus obtaining the number of bikes in the system. This results in a less precise prediction, since two different predictions are involved, but with a large enough time-step it is still possible to reach feasible solutions. The obtained results are presented in Table 4.16, with an example figure attach to showcase the prediction (Fig. 4.39).

Table 4.16 City – level error results from 01/01/2019 to 15/01/2019 (in – out demand).

	Random Forest	Gradient Boosting	Neural Network	
Overall	Avg. Bikes	303,86	303,86	303,86
	Avg. Error	73,903	141,169	136,105
	Acc. (%)	75,68	53,54	55,21
	Max. Error	382,413	468,774	503,3
	RMSLE	0,14	0,24	0,188
	RMSE	0,273	0,461	0,452

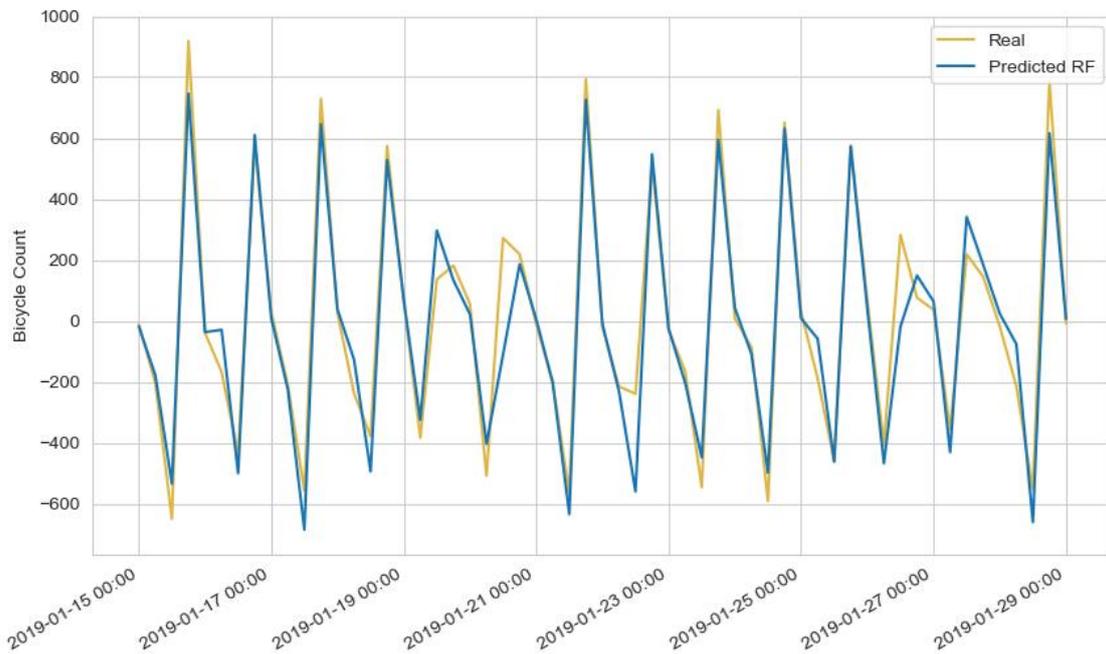


Fig. 4.39 City – level equilibrium prediction from 15/01/2019 to 29/01/2019 using Random Forest

We can observe that the highest accuracy is obtained by the Random Forest, almost 25% more compared with the other two algorithms for this particular case. In general, more accuracy is reached in the case of London compared to New York for this time period, mainly because the weather conditions were more stable in this case.

4.2.3 Cluster – level prediction

London’s case has six major clusters the case we will have that same number of models (Fig. 4.40). Each model will be able to predict the aggregated demand of each one of the sub-clusters included in the particular model. The minimum time-step considered in this case is 30min, with the results showcased in Table 4.17

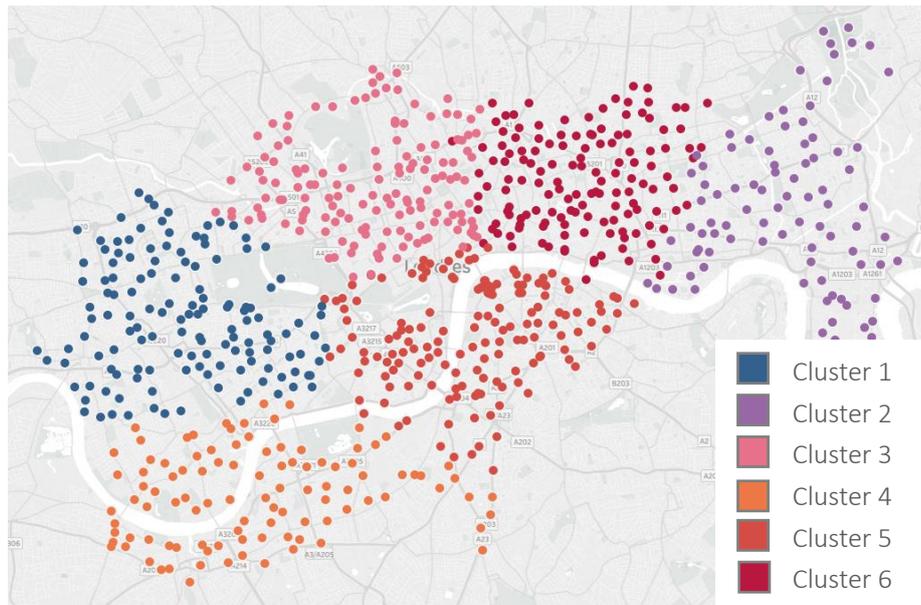


Fig. 4.40 Clusters used for London.

Table 4.17 Overall precision results for a cluster-level prediction of London City.

Cluster		1	2	3	4	5	6
Stations		136	96	144	102	154	141
Random	Avg. Bikes	18,441	9,121	20,287	8,401	23,164	23,638
	Avg. Error	6,036	3,449	6,463	3,177	7,516	7,098
	Acc. (%)	67,26	62,18	68,14	62,18	67,55	69,97
	Max. Error	83,017	59,992	125,724	54,518	130,011	197,572
Gradient	Avg. Bikes	18,441	9,121	20,287	8,401	23,164	23,638
	Avg. Error	6,694	3,811	7,155	3,389	8,639	8,51
	Acc. (%)	63,7	58,22	64,73	59,66	62,71	64,0
	Max. Error	100,779	62,971	127,161	60,142	151,706	173,142
Neural	Avg. Bikes	18,441	9,121	20,287	8,401	23,164	23,638
	Avg. Error	5,881	3,383	6,266	3,118	7,442	6,984
	Acc. (%)	68,11	62,91	69,11	62,89	67,87	70,45
	Max. Error	90,041	52,012	129,717	52,441	126,736	147,51

From the obtained results, it can be seen that the Neural Network model yields the best results for each of the clusters, followed up by the Random Forest Model. Looking at the average number of bikes being used. This system seems to be more uniform than New York, but there still is a differentiation between high-demand clusters and low demand ones. Cluster 1, 3, 5 and 6 can be classified as HD ones, since the average number of bikes used every 30min is around 20 in every

case, and the accuracy of the prediction is around 68%. These correspond to the area closer to the city center, where there is clearly more activity taking place. On the other hand, clusters 2 and 4 can be considered to be LD ones, not reaching an average of 10 bikes being used in the considered time-step, and a lower accuracy close to 63%. These clusters are situated outskirts of the city, where demand is generally lower (Fig. 4.41).

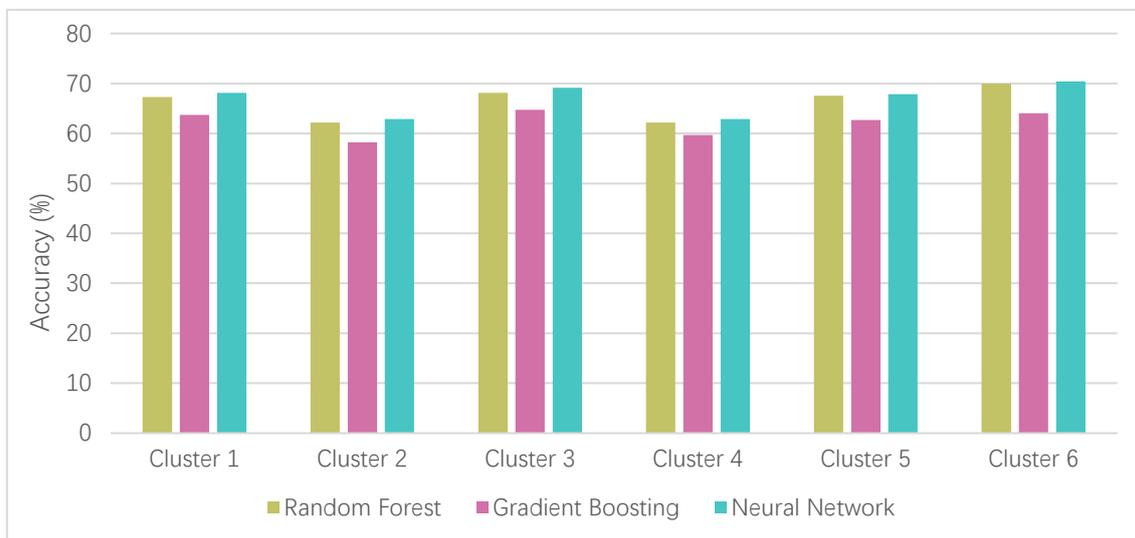


Fig. 4.41 Overall accuracy results for a cluster-level prediction of London City.

Since NN was able to reach the highest accuracy in all cases, the predictions showcased will only focus on this particular algorithm. The total amount of sub-clusters is 33, so only a selection of these will be analyzed into their full extent, selecting an example of a HD cluster and another example of a LD cluster. The dates predicted range from 15th January to 29th January.

The demand behavior of the HD cluster 5.4 is quite different from the others analyzed so far. In particular, there is a single peak for each day, one at 8am in the case of demand into the cluster (Fig. 4.42) and another one at 5pm for demand outside (Fig. 4.43), where there is usually another peak in the same type of predicted demand. Besides that, the magnitude of the peak hour compared to the other time steps is extremely different, which indicates these stations are mainly used just twice a day, with a clearly differentiated time pattern for demand in and out of the cluster.

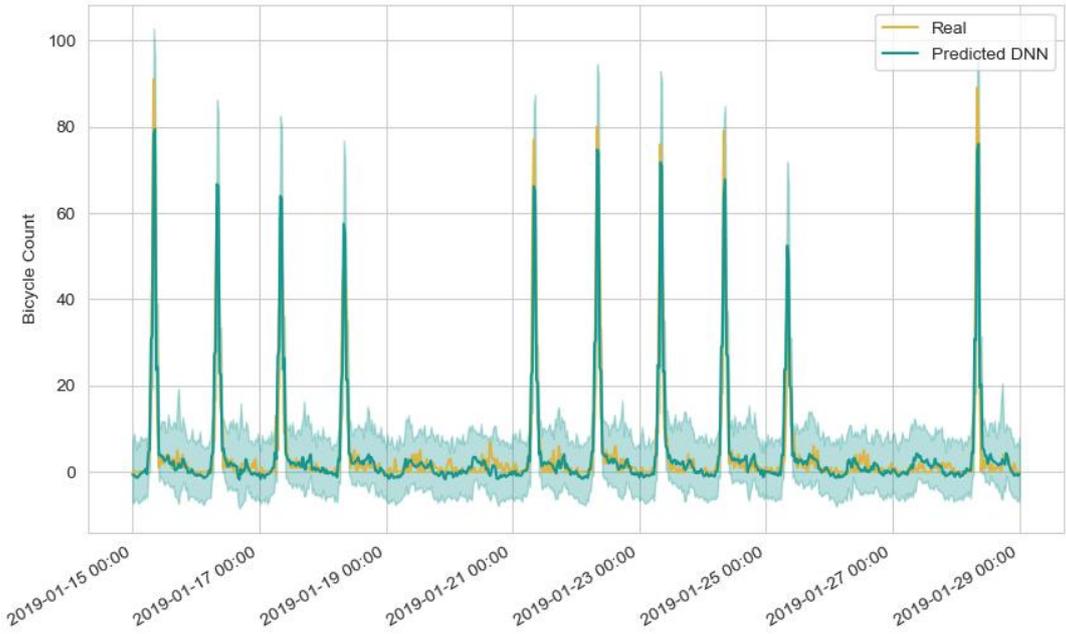


Fig. 4.42 HD Cluster demand prediction (In) from 15/01/2019 to 29/01/2019 using Neural Networks.

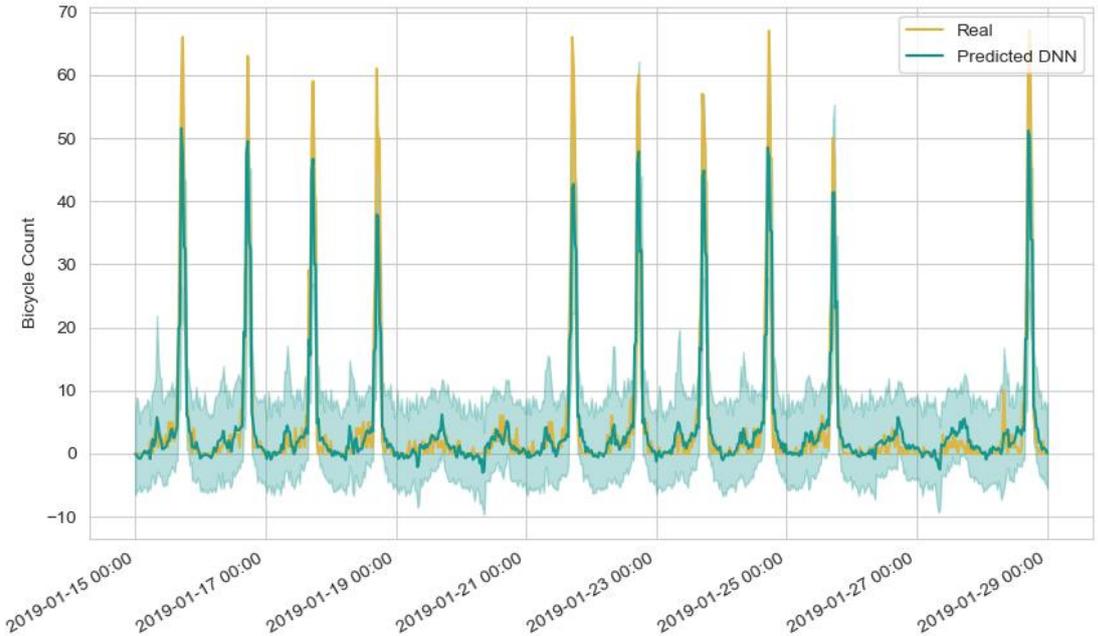


Fig. 4.43 HD Cluster demand prediction (Out) from 15/01/2019 to 29/01/2019 using Neural Networks.

Comparing the results with a LD cluster (Fig. 4.44, Fig. 4.45), we can see that the later have two clear peaks, as observed before in other cases. From the magnitude, the peak at 8am is higher in the demand out case (users going from the outskirts to the center to work), whereas the 5pm peak is more present in the demand in case (users getting back from work).

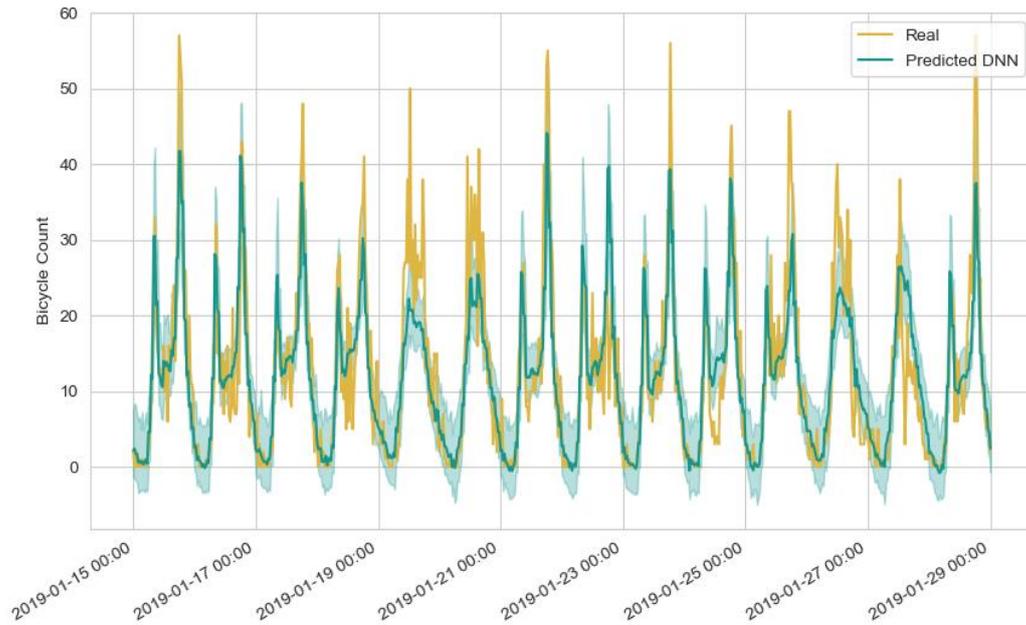


Fig. 4.44 LD Cluster demand prediction (In) from 15/01/2019 to 29/01/2019 using Neural Networks.

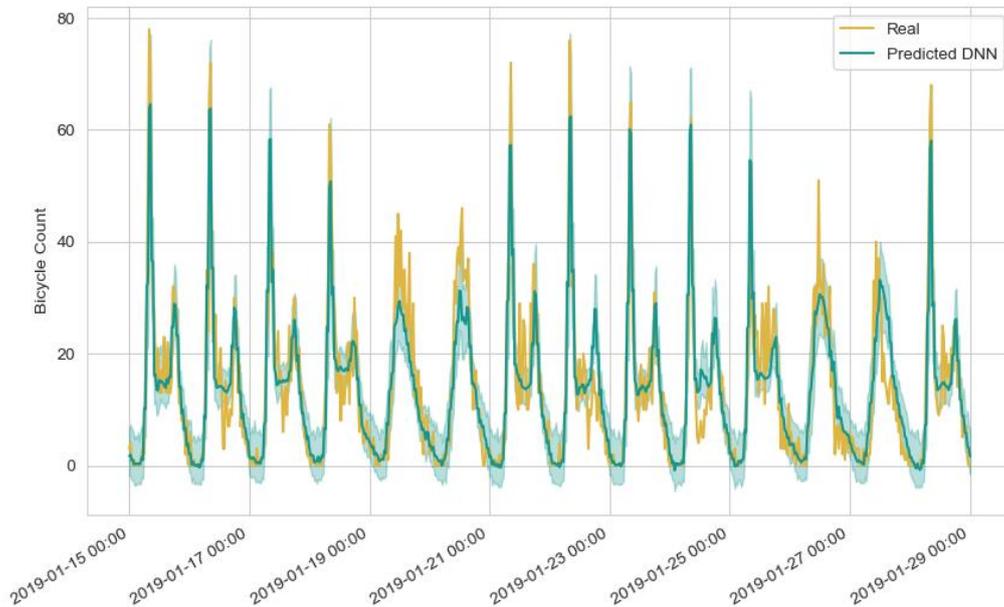


Fig. 4.45 LD Cluster demand prediction (Out) from 15/01/2019 to 29/01/2019 using Neural Networks.

From Table 4.18 we can see that the accuracy in the HD sub-cluster is actually lower than the LD one, which is a quite different result comparing with the case of New York City. This is mainly due to the demand shape and the considered formulation to define accuracy, since all the other error results (RMSLE, RMSE, average error), are lower in the HD sub-cluster rather than the LD one. Since all the demand in the HD sub-cluster is concentrated in a single time-step, the average number of bikes is much lower than, for example the LD case, which has influence in the calculation of the accuracy result. If we calculated the accuracy for the different time-steps (peak hour and non-peak-hour) we would obtain a higher accuracy (Table 4.19).

Table 4.18 Cluster– level error results from 15/01/2019 to 29/01/2019 with a 30min time-step with NN.

Cluster	Avg. Bikes	Avg. Error	Acc. (%)	Max. Error	RMSLE	RMSE
6.4 (HD)	4,493	2,276	49,36	28,793	0,083	0,131
1.1 (LD)	13,663	3,904	71,42	25,596	0,147	0,556

Table 4.19 Error results of the HD sub-cluster separated between peak and non-peak hours with NN.

Time	Avg. Bikes	Avg. Error	Acc. (%)	Max. Error	RMSLE	RMSE
Peak h.	63,500	9,984	85,85	21,719	0,146	0,320
Non-peak h.	5,915	3,387	42,74	27,832	0,098	0,126

4.2.4 Station – level prediction

The minimum time-step chosen for a station-level prediction is at least two hours and the showcased predictions correspond to a station from HD sub-cluster 6.2 and LD sub-cluster 1.1. The minimum time-step is changed from 1h to 2h since the accuracy with only this hour of difference increases dramatically (Fig. 4.46). This conclusion also matches the hypothesis taken into account to decide the time-step presented in Table 4.9, since the average number of bikes in an hour is between 1,5 and 7 in the HD cluster, whereas is smaller than 1,5 in the LD cluster, thus the chosen time-step is 3h in that case.

The HD sub-cluster 6.4 is not further analyzed since only a single station is included in it (station 14), this the provided analyzed in the previous section already corresponds to a station prediction. First, the error results for the three tested algorithms are showcased for the validation test (Table 4.20).

Based on the test set, the Neural Network algorithm obtains the best results in both cluster 6.2 (HD) and 1.1 (LD), whereas GBM exhibits the worst accuracy for the HD cluster, and RF the worst one for the LD one. Overall, NN seems to obtain the best results regardless of the situation.

Table 4.20 Station– level error results with a 2h time-step (HD) and 3h time-step (LD).

	Cluster	Avg. Bikes	Avg. Error	Acc. (%)	Max. Error	RMSLE	RMSE
Random Forest	6.2 (HD)	2,997	1,305	56,42	20,612	0,164	0,345
	1.1 (LD)	2,486	1,432	42,35	12,973	0,244	0,503
Gradient Boosting	6.2 (HD)	2,997	1,498	50,0	22,88	0,178	0,363
	1.1 (LD)	2,486	1,414	43,03	12,365	0,238	0,475
Neural Network	6.2 (HD)	2,997	1,241	58,56	18,21	0,155	0,319
	1.1 (LD)	2,486	1,324	46,63	12,09	0,226	0,452

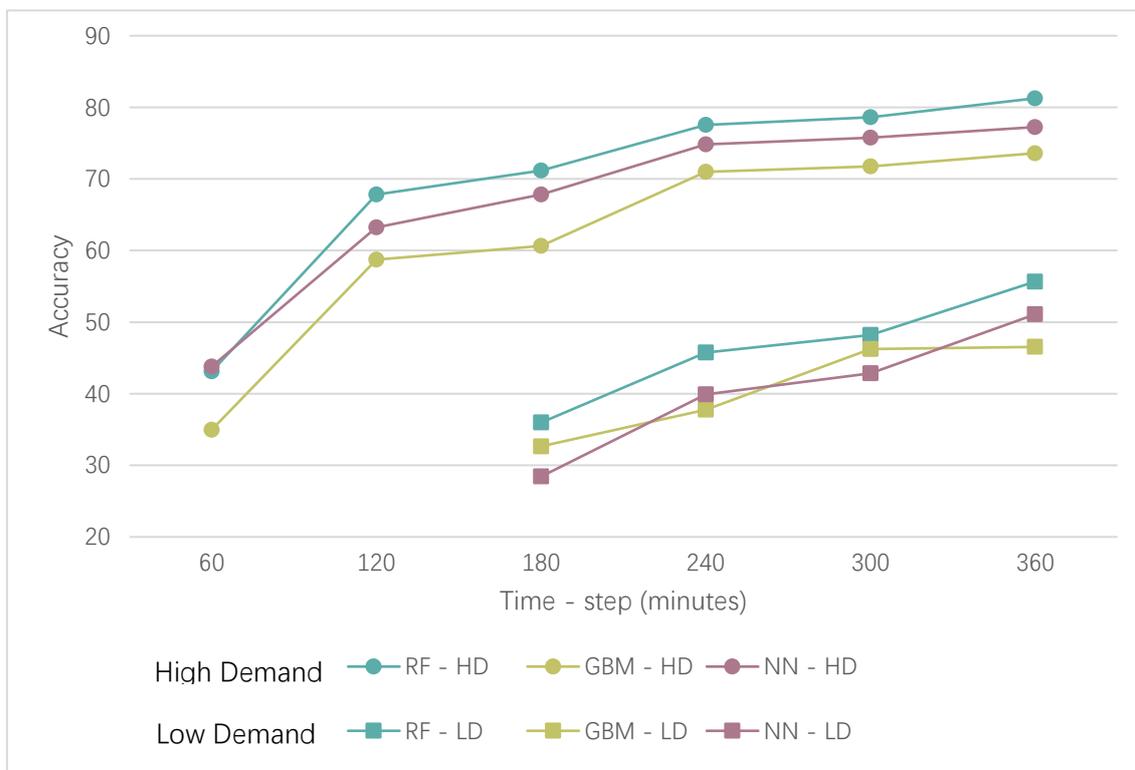


Fig. 4.46 Accuracy evolution when increasing the time-step interval for HD and LD stations.

In particular, for a prediction of the last two weeks of January (15th to 29th of January), we can obtain the error results for station 48 from the HD cluster and station 754 from the LD cluster. For the HD station, 1h time-step is chosen as the minimum one, but as for the LD one this is not enough as discussed before, so a 3h time-step is chosen instead.

From the obtained results in Table 4.21, one can see that the overall best results are obtained by the Random Forest Regressor, rather than the Neural Networks. As it was explained before, even though NN might yield the best results in most of the cases, this is not true for every case, as it was just demonstrated. RF not only obtains the best overall results, but also excels over the other algorithms in predicting the demand over working days, weekends or holidays and peak hours. It is only surpassed in the particular case for demand out predicted in by the NN, which its accuracy is slightly higher than the RF one. It is quite noticeable in Fig. 4.47 that the accuracy during weekends and holidays is way off, compared with the other results obtained. This might be because the actual weather conditions were different from the obtained data, since there might be errors sometimes as well. This error is noticeable in the prediction results from Fig. 4.48, Fig. 4.49 and Fig. 4.50.

As for the station's behavior, this particular dock has a slight disequilibrium, having more demand out of the station than into it. Since the average number of bikes going out of the station is higher, there is more information and thus the predicted results tend to be of higher quality than the predicted demand in the station.

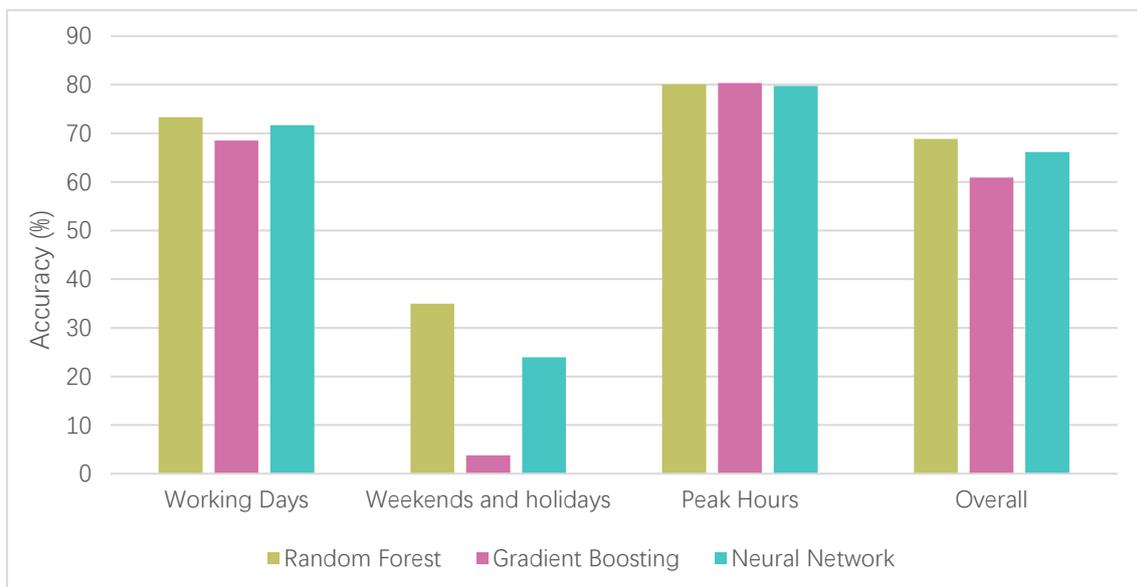


Fig. 4.47 Station 48 average accuracy results from 15/01/2019 to 29/01/2019 with a 2h time-step.

Table 4.21 Station 48 error results from 15/01/2019 to 29/01/2019 with a 2h time-step.

		Random Forest		Gradient Boosting		Neural Network	
Demand		In	Out	In	Out	In	Out
Working Days	Avg. Bikes	6,066	5,686	6,066	5,686	6,066	5,686
	Avg. Error	1,66	1,474	1,858	1,835	1,762	1,566
	Acc. (%)	72,63	74,07	69,36	67,73	70,96	72,46
	Max. Error	7,988	10,293	7,18	9,983	11,729	10,159
Weekends and	Avg. Bikes	2,417	1,646	2,417	1,646	2,417	1,646
	Avg. Error	1,429	1,166	2,165	1,692	1,616	1,402
	Acc. (%)	40,85	29,14	10,41	-2,82	33,11	14,82
	Max. Error	8,932	6,785	9,445	8,215	8,393	5,727
Peak Hours	Avg. Bikes	14,867	16,333	14,867	16,333	14,867	16,333
	Avg. Error	3,0	3,219	2,803	3,348	3,176	3,126
	Acc. (%)	79,82	80,29	81,14	79,5	78,63	80,86
	Max. Error	7,988	10,293	7,18	9,983	11,729	10,159
Overall	Avg. Bikes	5,03	4,538	5,03	4,538	5,03	4,538
	Avg. Error	1,595	1,387	1,945	1,794	1,72	1,519
	Acc. (%)	68,3	69,44	61,32	60,47	65,8	66,53
	Max. Error	8,932	10,293	9,445	9,983	11,729	10,159

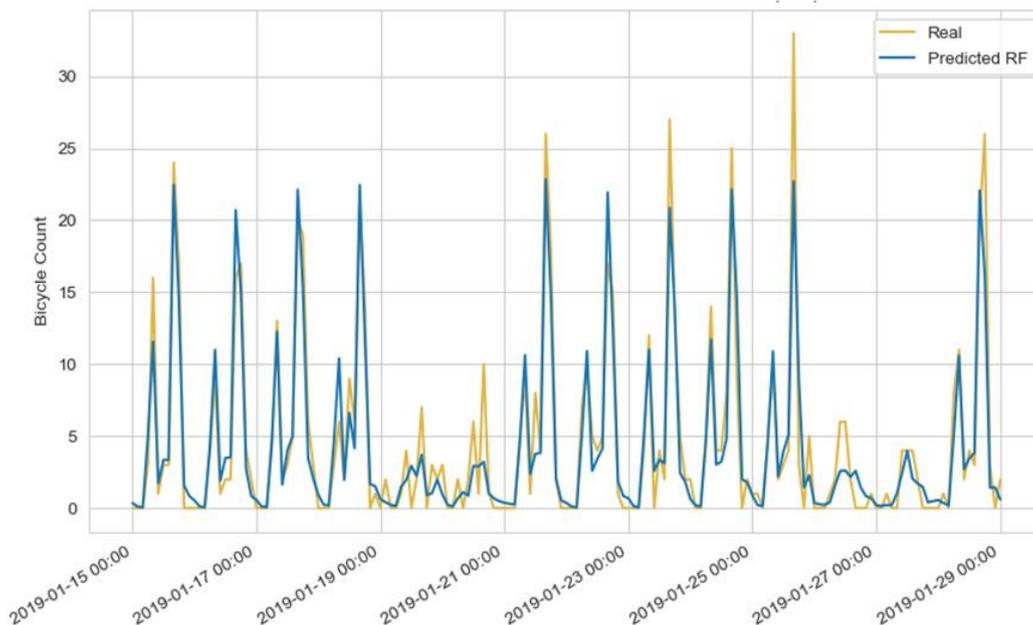


Fig. 4.48 Station 482 (HD) demand (out) prediction from 15/01/2019 to 29/01/2019 using RF.

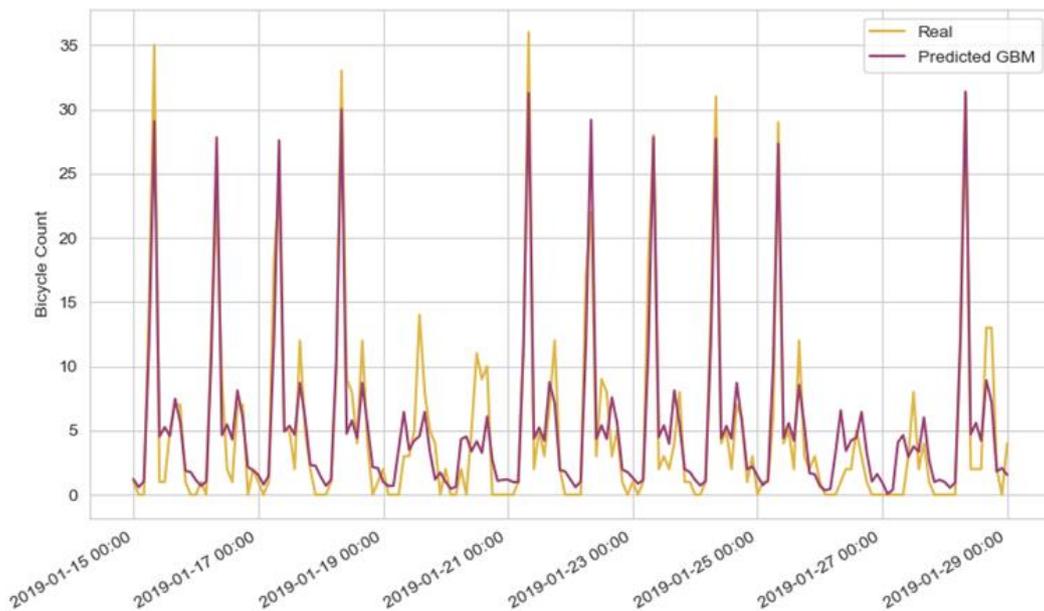


Fig. 4.49 Station 402 (HD) demand (in) prediction from 15/01/2019 to 29/01/2019 using GBM.

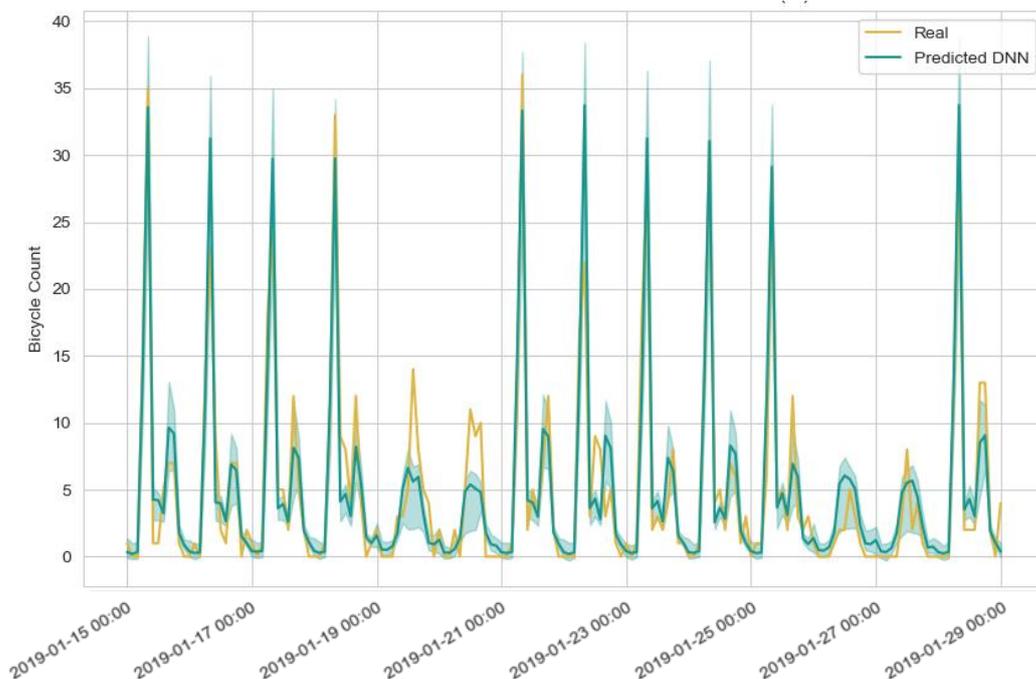


Fig. 4.50 Station 402 (HD) demand (in) prediction from 15/01/2019 to 29/01/2019 using NN.

Similarly, we can obtain the same results for station 754, corresponding to the LD cluster 1.1 (Table 4.22). From the obtained results, we clearly see the accuracy drop between a HD station and a LD, since now the maximum accuracy we can reach is 33%, whereas before we could reach up to 70% (Fig. 4.51). Overall, Random Forest keeps yielding the best results in this particular time-prediction and station, even though NN was yielding better results in the validation test. In case higher accuracy was needed, the maximum we can expect to reach is 58% when aggregating every 6h (Fig. 4.46). Again, some examples are showcased in Fig. 4.52, Fig. 4.53 and Fig. 4.54.

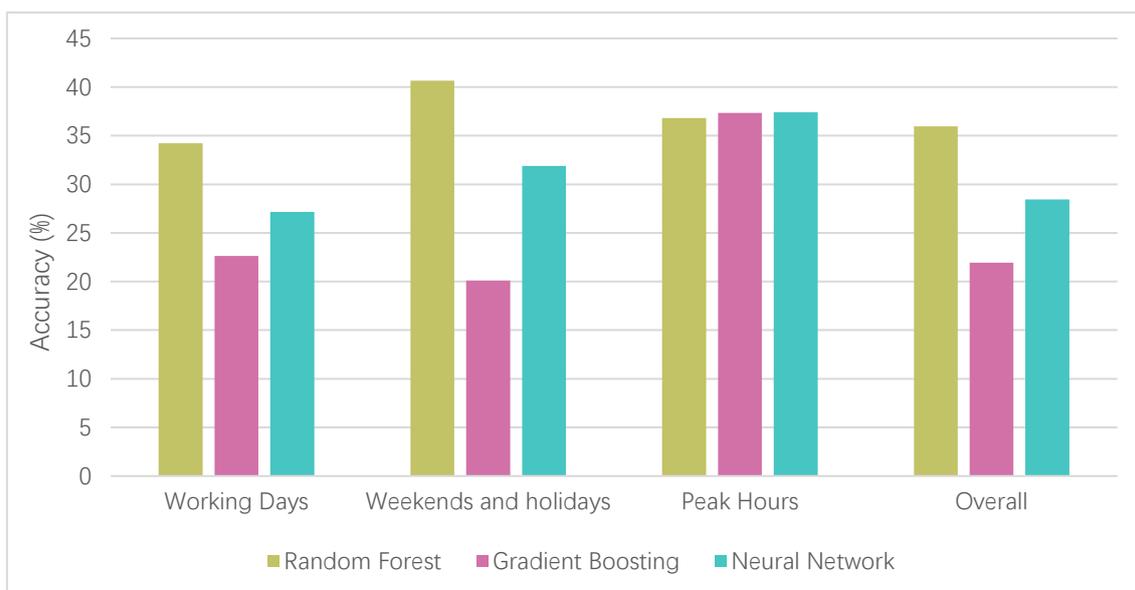


Fig. 4.51 Station 754 average accuracy results from 15/01/2019 to 29/01/2019 with a 3h time-step.

Table 4.22 Station 754 error results from 15/01/2019 to 29/01/2019 with a 3h time-step.

		Random Forest		Gradient Boosting		Neural Network		
		Demand	In	Out	In	Out	In	Out
Working Days	Avg. Bikes		1,111	1,136	1,111	1,136	1,111	1,136
	Avg. Error		0,751	0,726	0,808	0,931	0,791	0,847
	Acc. (%)		32,39	36,08	27,24	17,99	28,85	25,47
	Max. Error		4,298	3,126	4,521	3,842	4,593	3,715
Weekends and	Avg. Bikes		1,062	1,062	1,062	1,062	1,062	1,062
	Avg. Error		0,685	0,576	0,733	0,964	0,749	0,699
	Acc. (%)		35,54	45,79	30,98	9,24	29,55	34,22
	Max. Error		2,527	2,066	3,874	3,196	2,646	3,525
Peak Hours	Avg. Bikes		1,8	1,95	1,8	1,95	1,8	1,95
	Avg. Error		1,092	1,282	0,975	1,387	0,975	1,384
	Acc. (%)		39,32	34,28	45,81	28,87	45,81	29,03
	Max. Error		4,298	3,126	3,189	3,842	2,381	3,715
Overall	Avg. Bikes		1,097	1,115	1,097	1,115	1,097	1,115
	Avg. Error		0,732	0,684	0,787	0,941	0,779	0,805
	Acc. (%)		33,25	38,7	28,26	15,63	29,04	27,83
	Max. Error		4,298	3,126	4,521	3,842	4,593	3,715

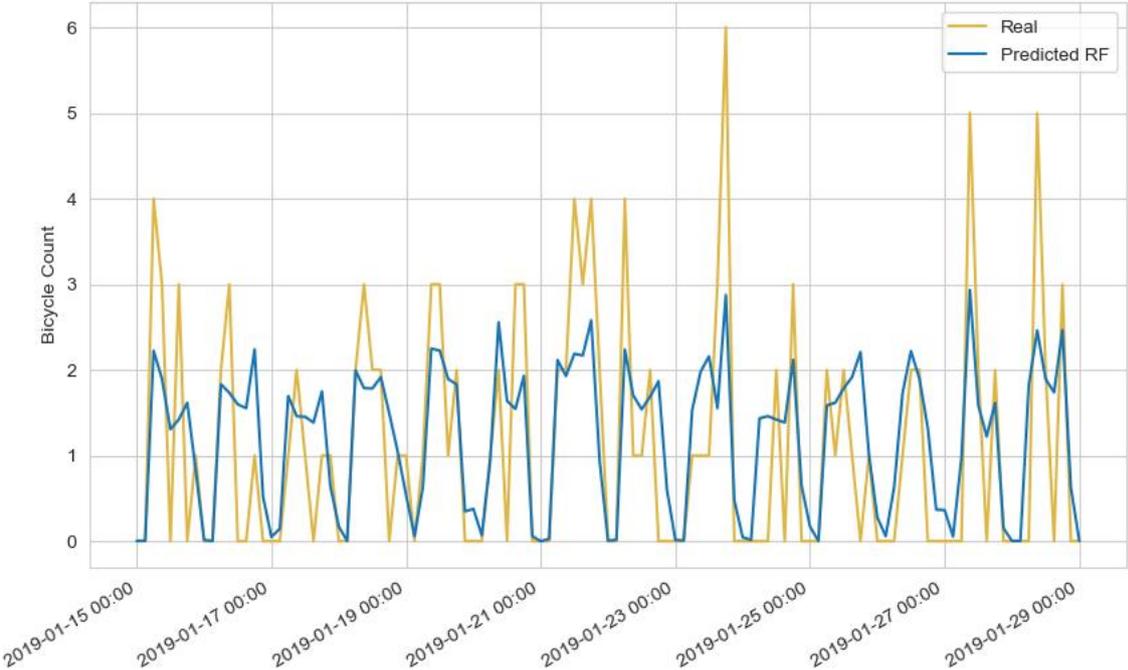


Fig. 4.52 Station 754 (LD) demand (in) prediction from 15/01/2019 to 29/01/2019 using RF.



Fig. 4.53 Station 754 (LD) demand (out) prediction from 15/01/2019 to 29/01/2019 using GBM.

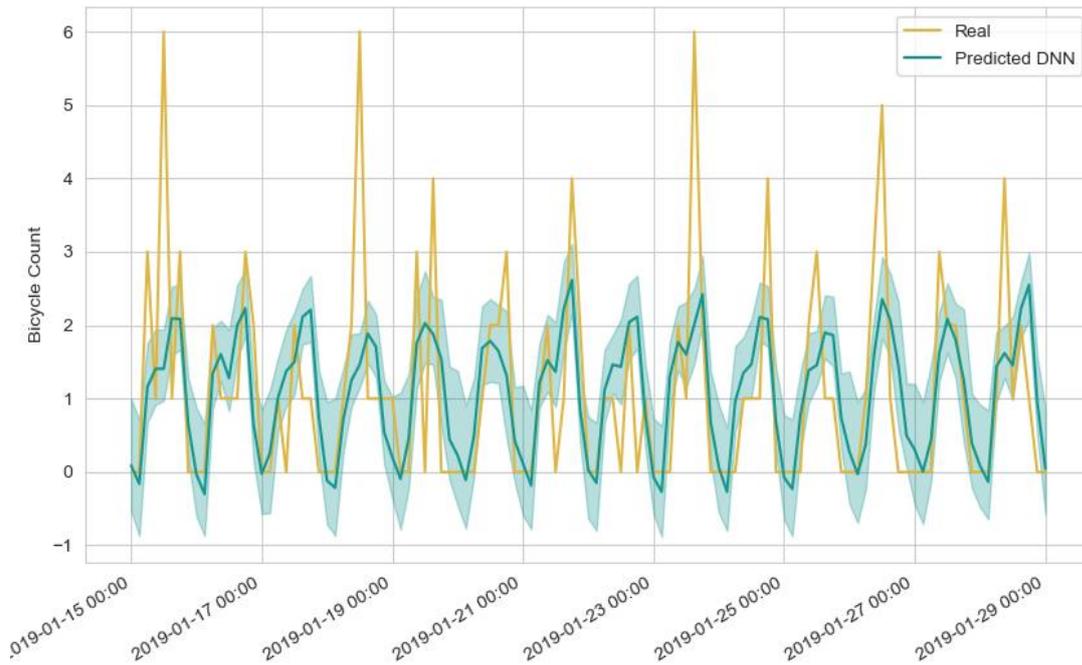


Fig. 4.54 Station 754 (LD) demand (out) prediction from 15/01/2019 to 29/01/2019 using NN.

One can also calculate the difference between the demand into the station and out of the station (clean demand), to try to predict the extra number of bikes in a station at a certain time interval. As it was mentioned before, this operation usually yields a lower accuracy than the obtained in the prediction of the demand itself. Thus, for this part, a time-step of 12h is chosen, to simulate the status of a station during midday and midnight, possible times for relocation to happen (Fig. 4.55).

It can be seen that most part of the error is concentrated where demand is higher, since the predicted volumes will be higher as well. As for the demand pattern, it seems that the algorithms tend to overestimate the most part of the demand, especially in the south – east part of the city. There is also a greater number of stations where the predicted clean demand is zero, when it should actually be around a single unit (either 1 or -1 depending on the station).

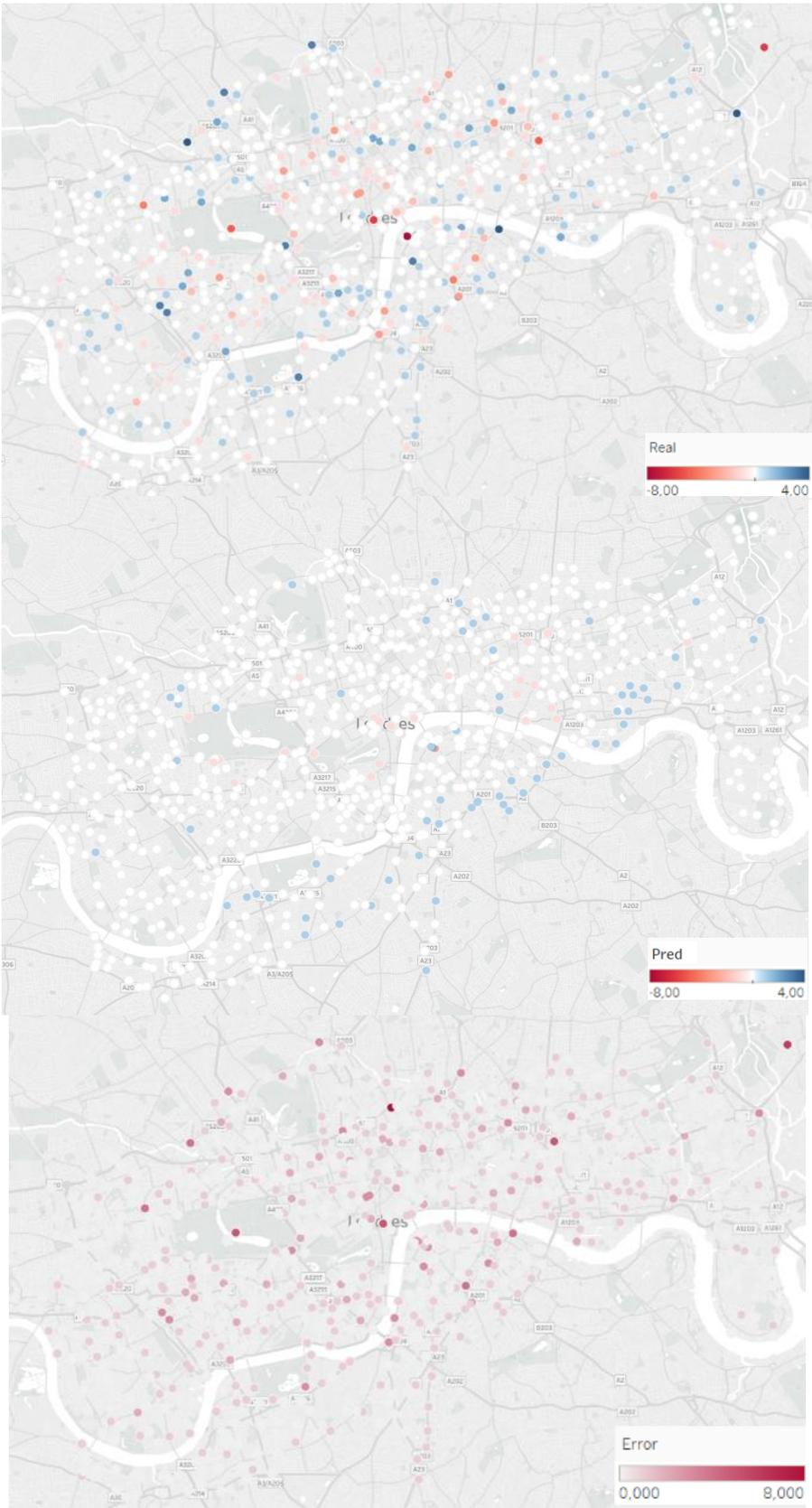


Fig. 4.55 Clean demand during the 15th of January. Real (top), predicted (middle) and error (bottom).

Finally, to test the robustness of the predictions against the uncertainty of the weather forecast, an error to the historical values has been applied to the temperature and wind registers. Rather than substitute the historic registers with actual predictions, the values have been added a percentage error as in Table 4.23. This approach is chosen so the actual error between the prediction and the real historic value is perfectly known, as well as it rather difficult to find historic data on predictions, rather than the actual historic data. The predictions have been carried out with the RF model, since it was the one that showed more precision in this particular study case, and for station 48, which pertained to the HD cluster.

Table 4.23 Prediction results for 15/01/2019 to 21/01/2019 with different weather percentage error.

	Error	0%	25%	50%	75%	100%
Temperature	Avg. Bikes	8,092	8,092	8,092	8,092	8,092
	Avg. Error	2,099	2,19	2,162	2,231	2,221
	Acc. (%)	74,06	72,94	73,28	72,43	72,55
	Max. Error	12,774	14,113	14,113	14,199	12,707
	RMSLE	0,215	0,238	0,229	0,228	0,214
	RMSE	0,366	0,389	0,386	0,388	0,394
Wind Speed	Avg. Bikes	8,092	8,092	8,092	8,092	8,092
	Avg. Error	2,099	2,123	2,069	2,125	2,207
	Acc. (%)	74,06	73,76	74,44	73,74	72,72
	Max. Error	12,774	14,543	12,761	14,415	13,416
	RMSLE	0,215	0,223	0,205	0,224	0,227
	RMSE	0,366	0,379	0,368	0,389	0,384

From the results, we can see that the model is quite robust against lack of precision in the weather forecast, at least for temperature and wind variables. The small percentage changes are rather due to rebuilding of the model (every time the model is rebuilt, the results can vary a small percentage) than to inaccuracies of the weather data. It should be noted that this is only true for these two variables, rather than for snow or rain, which have a great impact in the overall accuracy of the prediction.

4.3 Overall Results

From the case studies analyzed, not a single-algorithm was found to be optimal to reach the highest accuracy possible for every prediction level, city and time period. In particular, Random Forest and Neural Network yielded the best results overall, whereas Gradient Boosting did not reach the same level of precision. Considering the testing sets as the more explanatory ones, since the accuracy results from the validation test (January) might be quite dependable of the weather and use of the bikes on that day, the results for both case studies are showcased in Fig. 4.56.

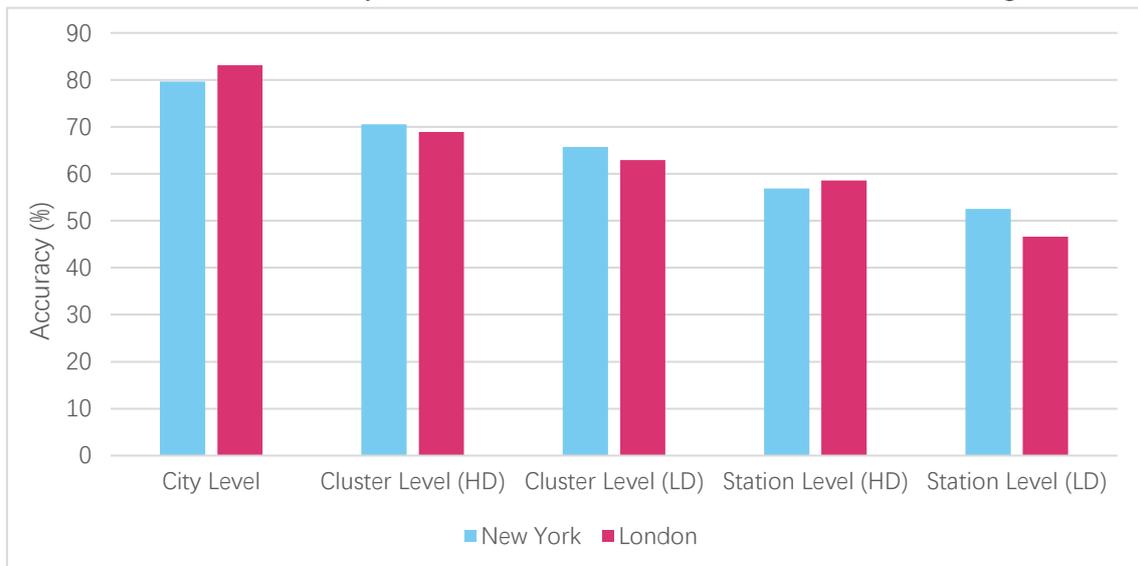


Fig. 4.56 Overall accuracy results for the case study of New York and London obtained via NN.

Considering only the test-set, the best results are always obtained when the proposed NN is applied. As mentioned before, this does not mean it will always yield the best results in every prediction period, but at least they will be close to the highest precision obtainable.

From the overall results, we can see that London case seems to have a better accuracy on a city level and HD station level, whereas New York yields better results in the other three cases. It should be noted that this comparison does not take into account same time-steps, but the smallest ones that yield a good enough accuracy. For example, in both cases the city level is predicted with a 15min time-step, whereas the both HD and LD clusters are predicted with a 30min interval. As for the HD stations, New York only requires 1h time-interval, whereas London need at least a 2h interval to reach feasible results. Similarly, for the LD stations New York used a 2h time-step, whereas London requires 3h. This makes it possible to say that, in general, New York is better predicted compared to London case, since the number of average bikes in each time-step is higher, which was shown to improve the accuracy of the results because more data was available.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

Throughout this master thesis, a novel method of demand forecasting in different aggregation levels of bike-sharing systems has been presented by means of machine learning algorithms. The proposed algorithms, Random Forest, Gradient Boosting, and Neural Networks, were optimized and applied in different prediction cases to test their behavior and adaptability on different environments, corresponding to different aggregation levels (city, cluster or station) or completely different bike-sharing systems.

By testing different aggregation levels, it was found that the used algorithms adapted well to the different cases. Higher accuracy was reached in the demand forecasting on a system level and cluster level, rather than on station level, where the variability was higher. In particular, an important difference was found between high-demand stations and low-demand ones, differentiated by the average amount of bikes being used in one hour. LD stations yielded less accuracy for the same time-step than HD ones, this a larger interval was proposed and justified. Since in the study case of New York City the number of bikes used was greater than London, the time-steps could generally be smaller and still yield the same level of precision.

As for the algorithms used, it was found that, in most cases, the Neural Network layout proposed yielded the best results on the testing set. Having said that, when the actual prediction was carried out for some weeks of January, it was demonstrated that Random Forest could still yield the least error nonetheless. Since the proposed NN system used a Bayesian approach, rather than just giving the output, this algorithm is still the recommended one to use, since more information is given to the user rather than a single output, such as a confidence interval.

The algorithms were also tested against weather forecast errors, exhibiting a great robustness for errors in temperature and wind variables due to the discretization method applied. As for precipitation (rain and snow), one should be careful when taking them into account, since they have a great impact in the model behavior.

5.2 Future Work

As future research, several more study cases should be carried out in different bike-sharing systems to get a deeper understanding of the relationship between the data input and the predicted demand. In particular, analyzing different systems might help understand the underlying relationship the demand in a certain station and the service-level (empty, full) of the nearby stations, which has been difficult to predict in the current study.

Moreover, possible changes on the variables used should be studied. For example, instead of the days of the week and holiday variables to be independent from each other, variables explaining the position of a working day between holiday could be used (working day between working days, working day between holiday, holiday between working day...). In that line, the number of categories to discretize the variables should also be considered, as well as the total amount of data. In this case, a whole year of data has been used to predict the next year, but it is possible than with less amount the accuracy will not decrease.

Finally, other algorithms could be tested instead of the three proposed ones. In particular, LSTMRNN (Long Short-Term Memory Recurrent Neural Network) has been shown to have potential in other studies on demand prediction of bike-sharing systems.

References

- [1] J. Winslow y O. Mont, «Bicycle Sharing: Sustainable Value Creation and Institutionalisation Strategies in Barcelona,» *Sustainability*, vol. 11, n° 3, p. 728, 2019.
- [2] Z. Furness, *One Less Car: Bicycling and the politics of automobility*, Philadelphia: Temple University Press, 2010.
- [3] S. A. Shaheen, S. Guzman y H. Zhang, «Bikesharing in Europe, the Americas, and Asia: Past, Present, and Future,» *Transportation Research Record: Journal of the Transportation Research Board*, vol. 20, n° 2143, pp. 159 - 167, 2010.
- [4] S. B. Jensen, «Free City Bike Schemes,» de *Conference Proceedings*, Copenhagen, 2000.
- [5] T. R. L. (. o. S. Transportation Research Group y . U. o. P. , «Monitoring and evaluation of the Bikeabout scheme in Portsmouth,» Transport Research Foundation Group of Companies, Berkshire, 1999.
- [6] Y. Sun, «Sharing and Riding: How the Dockless Bike Sharing Scheme in China Shapes the City,» *Urban Science*, vol. 2, n° 3, p. 68, 2018.
- [7] S. Schmöller y K. Bogenberger, «Analyzing External Factors on the Spatial and Temporal Demand of Car Sharing Systems,» *Procedia - Social and Behavioral Sciences*, n° 111, pp. 8 - 17, 2014.
- [8] H. Si, J.-g. Shi , G. Wu, J. Chen and . X. Zhao, "Mapping the bike sharing research published from 2010 to 2018: A scientometric review," *Journal of Cleaner Production*, no. 213, pp. 414 - 427, 2019.
- [9] L.-Y. Qiu y L.-Y. He, «Bike Sharing and the Economy, the Environment, and Health-Related Externalities,» *Sustainability*, vol. 4, n° 10, pp. 1 - 10, 2018.
- [10] J.-g. Shi, H. Si, G. Wu, Y. Su y J. Lan, «Critical Factors to Achieve Dockless Bike-Sharing Sustainability in China: A Stakeholder-Oriented Network Perspective,» *Sustainability*, vol. 6, n° 10, p. 2090, 2018.
- [11] Y. Feng, R. C. Affonso y M. Zolghadri, «Analysis of bike sharing system by clustering: the V'elib' case,» *International Federation of Automatic Control*, vol. 50, n° 1, pp. 12422 - 12427, 2017.
- [12] X. Zhou, «Understanding Spatiotemporal Patterns of Biking Behavior by Analyzing Massive Bike Sharing Data in Chicago,» *PLOS ONE*, vol. 10, n° 10, 2015.
- [13] Z. Gu, Y. Zhu, Y. Zhang, W. Zhou y Y. Chen, «Heuristic Bike Optimization Algorithm to Improve Usage Efficiency of the Station-Free Bike Sharing System in Shenzhen, China,» *International Journal of Geo-Information*, vol. 8, n° 5, p. 239, 2019.
- [14] Z. Kou y H. Cai, «Understanding bike sharing travel patterns: An analysis of trip data from eight cities,» *Physica A*, n° 515, pp. 785 - 787, 2019.
- [15] A. Sarkar, N. Lathia y C. Mascolo, «Comparing Cities' Cycling Patterns Using Online

- Shared Bicycle Maps,» *Transportation*, vol. 4, n° 42, pp. 541 - 559, 2015.
- [16] J. Warrington, C. Ruch y M. Morari, «Rule-based price control for bike sharing systems,» de *European Control Conference*, Strasbourg, 2014.
- [17] S. Ban y K. H. Hyun, «Designing a User Participation-Based Bike Rebalancing Service,» *Sustainability*, vol. 11, n° 8, pp. 1 - 14, 2019.
- [18] S. Reiss y K. Bogenberger, «A Relocation Strategy for Munich's Bike Sharing System: Combining an operator-based and a user-based Scheme,» *Transportation Research Procedia*, n° 22, pp. 105 - 114, 2017.
- [19] D. Pisinger , *Algorithms for Knapsack Problems*, Copenhagen: University of Copenhagen, 1995.
- [20] E. L. Lawler y J. K. Lenstra , *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, New Jersey: John Wiley & Sons, 1985.
- [21] J. Brinkmann, M. W. Ulmer y D. C. Mattfeld, «Inventory Routing for Bike Sharing Systems,» *Transportation Research Procedia*, vol. 19, n° 1, p. 203 – 210 , 2013.
- [22] L. Caggiani y M. Ottomanelli, «A modular soft computing based method for vehicles repositioning in bike-sharing systems,» *Procedia - Social and Behavioral Sciences*, n° 54, p. 675 – 684 , 2012.
- [23] L. Caggiana y M. Ottomanellia, «A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems,» *Procedia - Social and Behavioral Sciences*, vol. 87, n° 1, p. 203 – 210, 2013.
- [24] B. A. Neumann-Saavedra, P. Vogel y D. C. Mattfeld, «Anticipatory service network design of bike sharing systems,» *Transportation Research Procedia*, vol. 10, n° 1, p. 355 – 363, 2015.
- [25] J. Liu, L. Sun, W. Chen y H. Xiong, «Rebalancing Bike Sharing Systems: A Multi-source Data Smart Optimization,» de *International Conference on Knowledge Discovery and Data Mining*, San Francisco, 2016.
- [26] T. K. Ho, «Random Decision Forests,» de *3rd International Conference on Document Analysis and Recognition*, Montreal, 1995.
- [27] L. Breiman, «Random Forests,» *Machine Learning*, vol. 45, n° 1, pp. 5 - 32, 2001.
- [28] J. H. Friedman, «Greedy Function Approximation: A Gradient Boosting Machine,» Institute of Mathematical Statistics, Stanford, 1999.
- [29] K. Gurney, *An Introduction to Neural Networks*, Bristol: Taylor & Francis, Inc, 1997.
- [30] P. Norvig y S. Russell , *Artificial Intelligence: A Modern Approach*, Harlow: Pearson, 2009.
- [31] L. Pan, Q. Cai, Z. Fang, P. Tang y L. Huang, «A Deep Reinforcement Learning Framework for Rebalancing Dockless Bike Sharing Systems,» *arXiv: Artificial Intelligence*, vol. 1802.04592, 2018.
- [32] K. Gebhart y R. B. Noland , «The Impact of weather conditions on Bikeshare trips in Washington, D. C.,» *Transportation*, vol. 41, n° 6, p. 1205–1225, 2014.
- [33] C. Xu, J. Ji y P. Liu, «The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets,» *Transportation Research Part C*, vol. 95, n° 1, pp. 47 -

60, 2018.

- [34] I. Frade y A. Ribeiro, «Bicycle sharing systems demand,» *Procedia - Social and Behavioral Sciences*, vol. 111, nº 1, pp. 518 - 527, 2014.
- [35] H. Xu, J. Ying, F. Lin y Y. Yuan, «Station Segmentation with an Improved K-Means Algorithm for Hangzhou Public Bicycle System,» *JSW*, vol. 8, nº 9, pp. 2289-2296, 2013.
- [36] D. Singhvi, S. Singhvi, P. I. Frazier, S. G. Henderson, E. O' Mahony, D. B. Shmoys y D. B. Woodard, «Predicting Bike Usage for New York City's Bike Sharing System,» de *AAAI Workshop*, Austin, 2015.
- [37] M. Rosvall y C. T. Bergstrom, «Maps of random walks on complex networks reveal community structure,» *PNAS*, vol. 105, nº 4, pp. 1118-1123, 2008.
- [38] C. Thirumalai y R. Koppuravuri, «Bike Sharing Prediction using Deep Neural Networks,» *International Journal on Informatics Visualization*, vol. 1, nº 3, p. 10.30630, 2017.
- [39] Y. Pan, R. Chen Zheng, J. Zhang y X. Yao, «Predicting bike sharing demand using recurrent neural networks,» *Procedia Computer Science*, vol. 147, nº 1, p. 562-566, 2016.
- [40] J. Froehlich, J. Neumann y N. Oliver, «Sensing and Predicting the Pulse of the City through shared bicycling,» de *21st International Joint Conference on Artificial Intelligence*, Pasadena, 2009.
- [41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» *Journal of Machine Learning Research*, vol. 15, nº 1, pp. 1929-1958, 2014.
- [42] Y. Gal y Z. Ghahramani, «Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,» de *33rd International Conference on Machine Learning*, New York, 2016.
- [43] "Citi Bike," [Online]. Available: <https://www.citibikenyc.com/>. [Accessed 1 June 2019].