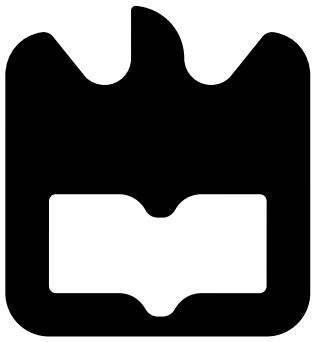
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2019

Diogo Macedo de Sousa

Decision Support Service for Bewegen Bike-Sharing Systems

Serviço de Suporte de Decisão para os Sistemas de Bike-Sharing da Bewegen



Diogo Macedo de Sousa

Decision Support Service for Bewegen Bike-Sharing Systems

Serviço de Suporte de Decisão para os Sistemas de Bike-Sharing da Bewegen

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia de Computadores e telemática realizada sob a orientação científica da Doutora Petia Georgieva, Professora Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho a todas as pessoas que acreditaram em mim, em especial aos meus pais por todo o apoio que me deram.

o júri / the jury

presidente / president	**Professor Doutor Luís Filipe de Seabra Lopes,** Professor Associado, Universidade de Aveiro
vogais / examiners committee	**Professora Doutora Catarina Helena Branco Simões Silva** Professora Auxiliar, Escola Superior de Tecnologia e Gestão de Leiria (Arguente Principal)
	Professora Doutora Petia Georgieva Georgieva Professora Auxiliar, Universidade de Aveiro (Orientadora)

agradecimentos
acknowledgements

À minha orientadora, Professora Petia Georgieva, pela disponibilidade, pelos conhecimentos transmitidos e pela motivação que me deu.
À Bewegen e Wegoshare que me permitiram avançar com este tema de dissertação disponibilizando as condições e a sua ajuda sempre que precisei.
Ao meu "mentor" na Wegoshare, Ricardo Costa, pelo acompanhamento e motivação transmitida.
Aos meus colegas na Wegoshare pelo constante apoio e paciência por me estarem sempre a ouvir falar da dissertação.
A todas as pessoas que fizeram parte destes últimos 6 anos da minha vida.
Aos meus pais, por todo o apoio. Sem eles eu não seria o que sou hoje.
Ao resto da minha família por terem estado sempre presentes e terem sempre palavras de incentivo que me ajudaram imenso.

Obrigado a todos, sem vocês este trabalho não teria sido possível.

Palavras-Chave

bike-sharing, aprendizagem automática, seleção de features, visualização de features, redes neuronais, k-nearest neighbour.

Resumo

Os sistemas de *bike-sharing* estão a tornar-se cada vez mais populares e a sua gestão mais complexa. O objetivo principal desta dissertação é o desenvolvimento de um serviço de suporte de decisão, baseado em métodos de aprendizagem automática, para os sistemas de bike-sharing da empresa Bewegen. Um objetivo secundário é o desenvolvimento de um mecanismo de recolha sistemática de dados de utilização do sistema, necessários ao desenvolvimento e teste dos métodos de aprendizagem automática. O serviço de suporte de decisão tem dois objetivos. O primeiro objetivo é a previsão do número de bicicletas em cada estação com 30 minutos de antecedência, informação esta a disponilizar aos clientes do sistema de *bike-sharing*. O segundo objetivo é a previsão do número de bicicletas em cada estação com 24 horas de antecedência, informação esta a disponilizar aos operadores do sistema no planeamento da distribuição das bicicletas pelas diferentes estações. Para cumprir com estes objetivos foram implementados dois algoritmos de aprendizagem automática: uma rede neuronal e um algoritmo *k-nearest neighbour*. Os testes realizados mostram que os algoritmos baseados em redes neuronais obtêm melhor desempenho nos dois objectivos. Os dados utilizados nos testes dos dois algoritmos são os dados históricos de um dos sistemas da Bewegen recolhidos desde 1 de janeiro de 2019 até 30 de abril de 2019.

Keywords bike-sharing, machine learning, feature selection, feature visualisation, neural networks, k-nearest neighbour.

Abstract Bike-sharing systems (BSS) are becoming very popular and, consequently, their management is becoming more complex. The main objective of this dissertation is the development of a decision support service for Bewegen bike-sharing systems applying machine learning (ML) methods. An additional objective is the development of an appropriate mechanism for systematic data collection, required in the development and test of the ML methods. The decision support service has two goals. The first goal is the prediction of the number of bikes in each station 30 minutes ahead of time, to be provided to the bike-sharing system clients. The second goal is the prediction of the number of bikes in each station 24 hours ahead of time, to be provided to the bike-sharing operators when deciding how to redistribute bikes among the different stations. In order to reach these two goals, two ML approaches were implemented: a neural network (NN) model and a k-nearest neighbour (k-NN) algorithm. The tests have shown that the NN algorithms provide better prediction results on both goals. The prediction algorithms were trained and tested with collected historical data from one of the Bewegen's BSS from 1 of January, 2019 until 30 of April, 2019.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A Bike-Sharing System (BSS) is a system in which bikes are rented by clients on a short-term basis for a cost that can be a combination of a periodical subscription and a rental time-based price. A BSS is composed by multiple stations, which are places where bikes are locked and can be released by proper client authentication. Rental time starts when the client unlocks the bike on a station and terminates when the client locks it again on a station. Clients can pick a bike from any station and return it to the same or any other station.

Public bike-sharing is becoming increasingly popular in the last decade with systems installed all over the world. The Canadian company Bewegen is a bike-sharing company with many different systems in North America, Central America and Europe. Due to the increased complexity of bike-sharing daily management and the need to improve the quality of the service provided to clients, human based management is not enough and advanced decision support systems must be envisioned. One possible approach is the one adopted in this dissertation of exploiting machine learning techniques that can provide useful information based on the historical behaviour of the system.

The information provided by machine learning techniques can be useful in different contexts. In this dissertation, two contexts are taken into consideration. First, a BSS requires daily management decisions on how many bikes should be operating and how they must be distributed among all available stations. Second, clients need information on bike availability (or space availability) at nearby stations when deciding how to move to their next destination (or where to drop their currently rented bikes).

The development work included in this dissertation was conducted in Wegoshare, a portuguese IT company with an office in Aveiro, who is in charge of all IT developments for all Bewegen BSSs.

1.2 Objectives

The main objective of this dissertation is the development of a decision support service for Bewegen BSSs based on machine learning methods. The decision support service has two goals:

- The first goal is to assist the clients of the BSS when they need to know the bike/space availability at the nearby stations in a short period of time. So, this is a **short term prediction task** where the goal is to predict the number of bikes in each station 30 minutes ahead of time.
- The second goal is to assist the system operators on how they should distribute the bikes among all stations in the system. This is a **long term prediction task** because the operators need this information much early in advance. So, the goal is to predict the number of bikes in each station 24 hours ahead of time.

An additional objective of this dissertation is the development of an appropriate mechanism for systematic data collection to be plugged in Bewegen internal system, required in the development and test of the machine learning methods.

1.3 Dissertation Outline

The dissertation is organised in five chapters as follows:

- Chapter 1 provides an introduction to this dissertation.
- Chapter 2 introduces the concept of a BSS, describes the BSSs of the Bewegen company and reviews the scientific literature related to the prediction problem in hand.
- Chapter 3 describes all the work done to prepare data for machine learning algorithms, including data collecting, data processing, data visualisation, and feature description and ranking.
- Chapter 4 describes and evaluates the two machine learning approaches (neural network and k-nearest neighbour) taken in order to predict the number of bikes on each stations in Bewegen BSSs.
- Chapter 5 draws the main conclusions of the work described in this dissertation and discusses some possible future work.

Chapter 2

Bike-Sharing Systems - State of the Art

This section introduces the concept of bike-sharing system (BSS), describes the BSSs of the Bewegen company and reviews the scientific literature related to the prediction problem in hand. In section 2.1 the background of BSSs is described. In section 2.2 the Bewegen BSSs are shortly introduced. Section 2.3 describes some of the largest and most important BSSs worldwide. Section 2.4 discusses the major BSS characteristics that are typically considered as features in prediction tasks. Finally, in section 2.5 are reviewed the most frequently applied Machine Learning (ML) algorithms in bike-sharing predictions.

2.1 Background

A contemporary BSS is referred in [10] as a system that enables short-term rentals of bikes taken from one station and returned to the same or other station. Stations are places where bikes are locked and can be released by proper client authentication. There are two types of stations [11]. The first type are stations with docks for clients to park the rented bikes. For a long time, this type of stations was the only one used in bike-sharing. In recent years, a new type of stations has appeared, commonly named as dockless stations. In this type, clients can park the bikes where they want inside the station limits and there is no specific mechanism in the station (such as docks) to lock the bikes.

Rental time starts when the client picks up the bike at a certain station and terminates when the client returns the bike to the same or any other station. Stations are placed in strategical locations over the system limits in order to satisfy the clients needs.

Due to many factors, the use of BSSs has increased significantly over the last years. Among the most significant ones, these systems are an environmentally friendly solution for transportation, provide a much healthier way to move around in the city, reduce traffic congestion and fuel consumption and reduce the transportation costs within short distances. An interesting conclusion drawn in [12] is that BSSs have a positive financial impact in the Minneapolis businesses. The results show that food-related businesses and job destinations

are positively affected by the existence of nearby bike-sharing stations. Positive impacts in several businesses are also reported in [13] for the Washington DC BSS.

2.2 Bewegen Systems

Bewegen is a Canadian company that has BSSs all over the world. Table 2.1 presents the list of all current Bewegen systems identified by their locations (city and country). The table shows the number of stations and the number of bikes of each system, which spans from quite small systems (the smallest is in Guilford, United Kingdom, with only 2 stations and 20 bikes in total) to significantly large ones (the largest is in Tartu, Estonia, with 69 stations and 750 bikes).

Continent	Country	City	No Stations	No Bikes
Europe	Portugal	Vale do Lobo	5	30
		Coruche	4	27
		Lagoa	3	30
		Rio Maior	4	24
		Leiria	11	220
	United Kingdom	Guilford	2	20
		Forth Valley	10	100
	Estonia	Tartu	69	750
America	United States of America	Birmingham	40	400
		Richmond	22	220
		Howard County	8	80
		Summit County	10	130
		Raleigh	30	330
		Riverside	7	55
		Pioneer Valley	50	500
		Columbia	15	135
	Costa Rica	Peninsula de Papagayo	7	60

Table 2.1: Bewegen systems information

All Bewegen systems operate with two types of bikes: regular bikes and pedelecs. Pedelecs are electric bikes with a motor that helps riders to pedal in hard situations like hills or long distances. Moreover, both types of bikes have a GPS module and a sim card to report periodically their geographical location through a local available mobile network. Therefore, at any time, the system has the information of the location of each bike.

An important characteristic in both types is that each bike has a padlock. This is useful, for example, when a client in the middle of a ride needs to make a stop where no station is nearby. In this case, clients can just lock the bike (using the padlock) to some fixed structure and the bike is safely parked. When this happens (i.e., the bike being locked

with the padlock outside a station), the system is reported by the bike of this situation and considers that the bike is in a intermediate stop. This information is currently used only for usage statistics. Nevertheless, it might be used in the future when system expansion is needed, i.e., if new stations are required, their potential best locations can be determined by the more frequent intermediate stop locations.

The padlock is also useful when a station is full and a client wants to park its rented bike. In this case, the system lets the client to lock its bike with the padlock to the wheel of a docked bike. When bikes are locked in one of the station docks, it is named a Primary Lock. When the bikes are locked using the padlock, it is named Secondary Lock.

Bewegen systems operate with two different types of stations: physical stations and virtual stations. A physical station has docks where bikes can be picked or dropped. One of the main advantages of the docks is that when the bikes are docked, they are being charged and therefore the risk of draining all bike's battery in the next ride is significantly reduced. Another advantage is that the bikes are more securely locked and the risk of being stolen is minimised. On the other hand, a virtual station is an empty area in the map where the bike inside the area considers it as a station. Clients can park the bikes in the virtual stations but, because there are no docks, the bikes can only be dropped as Secondary Locks (i.e., locked with the padlock).

Bewegen systems enable two major ways of bike renting: bikes can be unlocked with smart cards through RFID technology, or with the mobile app of the system. A third way is described next but requires the existence of a kiosk on the station.

Kiosks are a bike-sharing component that exists only in some stations. Their purpose is to provide additional support to clients, as kiosks allow them to do different actions more easily. Their most important functionalities are:

1. Allow clients to register in the system as occasional users;

2. Allow clients to unlock a bike in the station;

3. Allow clients to buy single rides;

4. Provide information about the system to clients.

Figure 2.1 and Figure 2.2 show 2 examples of bikes from Bewegen BSSs parked in stations of Zyp Bikeshare System (Birmingham, United States of America) and U-Bike (Leiria, Portugal), respectively.

2.3 Major Bike-Sharing Systems

This section briefly describes some of the largest and most important BSSs worldwide.

Figure 2.1: Zyp Bikeshare System bikes

Figure 2.2: U-Bike Bike-share System bikes

2.3.1 Hangzhou Public Bicycle

Hangzhou Public Bicycle System is one of the first contemporary BSSs. It was launched on 1st of May 2008 with 2800 bikes and 60 stations. Currently it has 78000 bikes and 2965 stations. The first years of the system and its impact on the city of Hangzhou are described in [14]. The authors argue that BSSs such as *Hangzhou Public Bicycle* are not just a competitor to other public transportation but, actually, a complement to them.

This BSS is currently operated by the newly-formed Hangzhou Public Bicycle Development Company called Hangzhou Urban Public Bicycle Sharing Program and financed by the government. Figure 2.4 shows Hangzhou Public Bicycle bikes parked in one of the Hangzhou stations.

Figure 2.3: Hangzhou Public Bicycle bikes [1]

2.3.2 BIXI Montreal

BIXI Montreal is one of the largest BSSs worldwide and the first to be implemented in the american continent. It was lunched in 2009 and currently has 540 stations and 6200 bikes. The authors of [15] studied the first six years of the system existence. Since the beginning, the system expanded rapidly reaching its peak in 2012 and stabilised since then.

The system has been operated and further developed by Public Bike System Company (PBSC). PBSC started as a public company and, meanwhile, became a private company that started to offer bike-sharing expertise to many other systems around the world. Figure 2.4 shows *BIXI Montreal* bikes parked in one of Montreal stations.

Figure 2.4: BIXI Montreal bikes [2]

2.3.3 Capital Bike Share

Capital Bike Share is known as the BSS in Washington DC but, in fact, the system covers six jurisdictions according to their website: Washington DC; Arlington, VA; Alexandria, VA; Montgomery County, MD; Prince George's County, MD; and Fairfax County, VA. The system was launched in September of 2010, it has more than 4300 bikes and more than 500 stations, and was the largest BSS in the United States until Citi Bike Share opened in New York (Section 2.3.4).

Capital Bike Share uses the BIXI-branded system provided by Montreal-based PBSC Urban Solutions (Section 2.3.2) and is operated by a public-private partnership between all local governments. Figure 2.5 shows *Capital Bike Share* bikes parked in one of the Washington DC stations.

2.3.4 Citi Bike Share

Citi Bike Share, launched in 27 May 2013, is a privately owned public BSS serving the New York City, including Manhattan, Queens, Brooklyn and New Jersey city. Named after the lead sponsor Citigroup, it is operated by Motivate. The BSS bikes and stations use BIXI-branded technology from PBSC Urban Solutions. The last records show that this system has 739 stations and 10494 bikes. Figure 2.6 shows *Citi Bike Share* bikes parked in one of its stations.

Figure 2.5: Capital Bike Share bikes [3]

Figure 2.6: Citi Bikes bikes [4]

2.3.5 Vélib' Métropole

Vélib' Métropole is a large-scale public BSS in France. It operates in Paris and other 64 surrounding cities. It was launched on 1 January 2018 and has 1100 stations and 20000 bikes. The previous BSS in France was Vélib' system, lauched in 15 July 2007. It had 1229 stations and 14500 bikes and was one of the biggest systems in the world. This first system was closed on 31 December 2017 and replaced by *Vélib' Métropole*. Figure 2.3.5 shows *Vélib' Métropole* bikes parked in one of its stations.

Vélib' Métropole bikes [16]

2.4 Data Cleaning and Feature Extraction

The first step in building a BSS prediction model is to select the most relevant data (features) to serve as input to the prediction algorithms. The collected historical data of bike pickups and drop-offs is naturally the most important data used in all analysed research works. However, other features like temperature, rain, or day of the week are also considered in some of them. In this section are presented the features used in the reviewed references.

2.4.1 Historical data

The historical data describes the BSS dynamics in the past and usually includes information about the number of pickups and drop-offs at different time intervals, as well as the occupation of each station at different time instants. This data is then used by a number of statistical algorithms for prediction of the bike-sharing traffic. The most typical methods for BSS prediction based only on historical data are Historical Average (HA) and Auto-Regressive and Moving Average (ARMA) models.

Historical Average (HA) model

The HA model computes the average of the historical data over specified time windows (day, time) and locations (i.e. stations) to predict the future BSS behaviour [17]. Consider, for example, the prediction of the bike-sharing behaviour at a given station on Friday between 16h and 17h. The HA model proposed in [18] considers the pickups and drop-offs over all collected past Fridays between 16h and 17h from the same station. Alternatively, the HA model in [19] clusters the days in workdays and non-work days and for the same example, it takes into account the registered values of all past workdays between 16h and 17h from the same station. A different HA model is proposed in [20] where historical values between 16h and 17h from the same station, 1 day before, 2 days before, 3 days before, 7 days before and 14 days before, are considered in the prediction.

Auto-Regressive and Moving Average (ARMA) model

ARMA model is a well known approach for time series prediction. Since the pickups and drop-offs of BSS are time series, ARMA is also popular in bike-sharing predictions. In this approach, historical data of the last n hours is considered (n is defined for each particular case). For example, the ARMA model proposed in [21] makes predictions for the next n hours based on the previous n hours, i.e., if one wants to predict the bike-sharing traffic 3 hours ahead, the method uses the last 3 hours of historical data (in [21], the maximum value for n is 24 hours).

2.4.2 Weather

An important additional feature in BSS is the weather as it significantly affects the intensity of the bike usage. The most typically considered aspects of the weather are: the maximum, minimum and average temperature, the volume (and/or intensity) of the rain or snow and the speed of wind.

The authors of [15] take into account only two weather dimensions: the average temperature and the rain quantity (in millimetres). They demonstrate that the rain has a negative impact on the bike usage (as can be expected) while conditions of mild and not too high temperatures have a positive impact. The same conclusions are drawn in [5], with respect to the effect of the max daily temperature on the average daily NYC Citi Bike trips (see Figure 2.7).

More aspects of the weather are considered in [22], namely the daily maximum and minimum temperatures, the precipitation, the snow and the rainfall. Their statistical results are similar to the ones in [15], adding also the expected negative impact on the bike usage of snow conditions.

In [18] is demonstrated that the humidity, the visibility and the wind speed have also a significant influence on the number of pickups at the stations.

The authors of [23] link hourly bike usage data to more detailed hourly weather data, including temperature, rainfall, snow, wind, fog, and humidity levels. Their results show that among all considered weather aspects, cold temperatures, rain and high humidity levels have a dominant effect on the reduction of the bike-share usage and also on the decrease of the average duration of bike trips.

2.4.3 Time

The influence of the hour of the day on the number of pickups/drop-offs in the stations is studied in [18]. This study concludes that the system has a very low usage during the hours of the night and, as a consequence, they remove the data entries of the night hours. In Figure 2.8, from [5], it is shown the average number of trips in NYC Citi Bike as a function of the day and night hours. In this figure, it is easy to observe that both in weekdays and in the weekends the number of bike trips is much less between midnight and 7h in the morning.

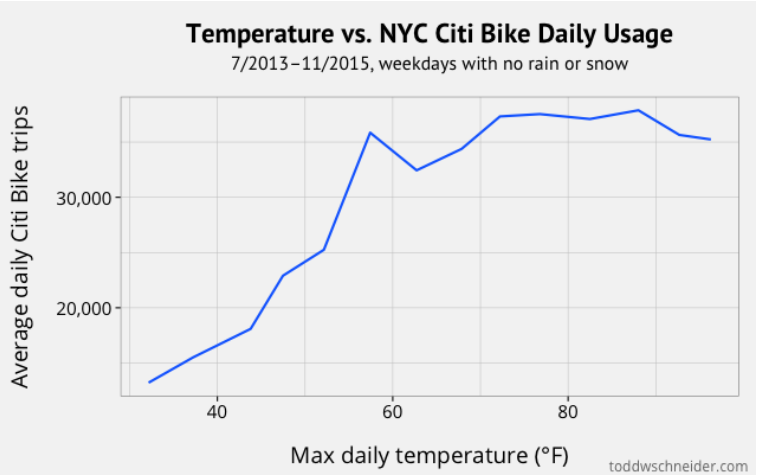


Figure 2.7: NYC bike trips vs. temperature [5]

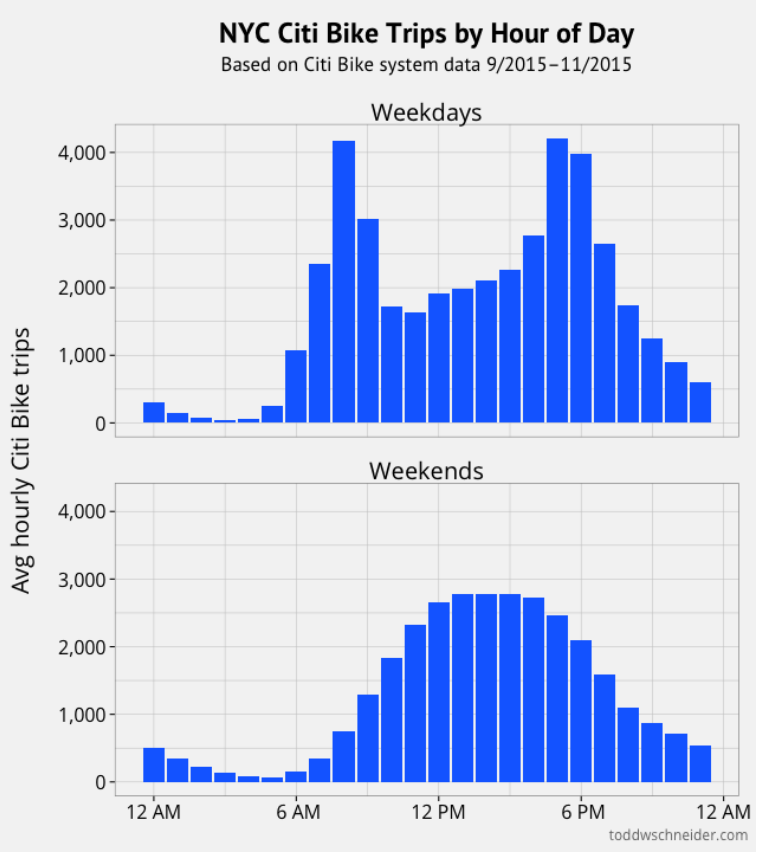


Figure 2.8: NYC bike-share usage per hour [5]

The seasons of the year (Winter, Spring, Summer and Autumn) also affect the system usage levels. There is much bigger demand during the Summer than during the Winter

season as shown, for example, in [24].

There are two types of BSSs with respect to the target clients. First, BSSs whose main business target are the tourists/visitors. In this type of systems, the stations are usually located in touristic places in the cities. Second, BSSs whose main business target are the local citizens. In this type of systems, the stations are usually located around job related areas and residential areas [18, 25]. An efficient way to identify the BSS case of study is by analysing the bike usage differences between week days, weekends and holidays. Normally, in a touristic oriented BSS, the bike traffic increases over the weekends and holidays. On the other hand, a BSS oriented to local citizens has more usage during week days, as it often serves as a mean of transportation to the work places. For example, the NYC Citi Bike Sharing is a more touristic oriented BSS since its usage increases 25% from work days to weekends (and holidays), as reported in [24].

2.4.4 Users

Very few research works make analysis of the clients personal information. For example, the Bixi Bike-share (Section 2.3.2) members are described as young people with much higher percentage of man than women [15]. The generations with more bike users are aged between 25 and 39 years old. However, none of the reviewed references considers the clients characteristics as BSS prediction features.

2.4.5 Stations

Bike-sharing predictions are normally applied separately on each station of a BSS. However, some studies propose to cluster the stations in groups in order to get more data for prediction. For example, [19] proposes to cluster the stations according to their geographical location and according to pickups/drop-offs traffic. In BSSs that are spread over more than one city, the weather conditions may not be the same in all locations. Therefore, station clustering by location can be a valuable factor as clusters may have different weather characteristics on the same moment. The authors of [25] add that by station clustering, small communities can be found where a lot of trips are being done inside each community. One example of a BSS spread over multiple small cities is ValleyBike, one of the Bewegen BSSs, shown in Figure 2.9.

It is shown in [22] a study of the station-level predictive performance where stations are divided in 5 clusters, depending on their locations and behaviour. On the other hand, paper [26] presents a different way to cluster stations based on their surrounding built environment.

Paper [19] points out that the dynamics of neighbour stations can influence the behaviour of each other. For example, if one station is full, the neighbour stations tend to have more drop-offs. On the other hand, if the station is empty, the neighbour stations tend to have more pickups. When using station clustering, this mutual influence between neighbour stations is lost when they are in the same cluster (as the information becomes

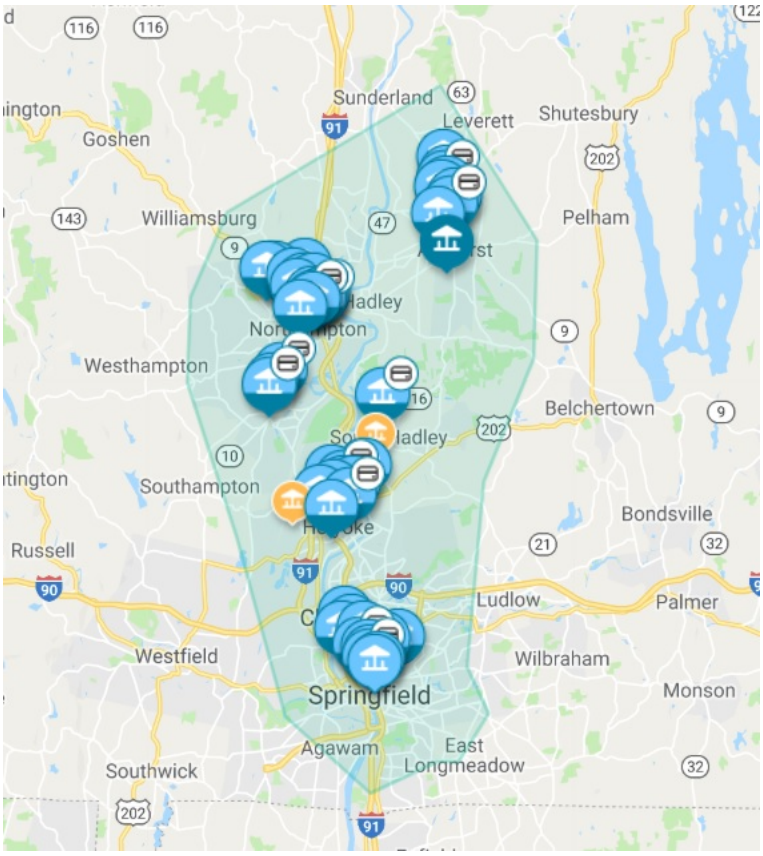


Figure 2.9: ValleyBike BSS map

aggregated for all stations of the same cluster) and this is one of the disadvantages of station clustering.

In paper [27], some conclusions are made about the impact of placing the stations in good locations. Certain locations are more suitable for bike-sharing stations, such as close to small business areas, to big universities (and schools) and to stations of other public transportation systems (such as train or bus).

2.4.6 Ranking features

Besides the selection of the features that influence the usage of a BSS, some research works also rank the features according to their influence on the historical data.

Paper [15] ranks the selected features in two different scenarios. When the aim is to predict the number of pickups/drop-offs per day, the proposed ranking considers only 3 features in the following order: the first, and more important, is the historic number of bikes in the stations, the second is the average temperature and the last is the precipitation. When the aim is to predict the number of pickups/drop-offs per day and per station, the proposed ranking considers much more features and in the following order: the first is the

capacity of the stations, the second is the average temperature, the third is the membership pricing plan, the fourth is the population density, the fifth is the number of buses passing within 500 meters of the station and the last one is the precipitation.

In paper [21], the features are divided into four categories ranked by their importance. The Very Relevant category has only the number of bikes in each station in the last 1 hour time interval. The Relevant category has the temperature and the number of bikes in a sequence of 1 hour time intervals prior to the last one. The Average category contains the weather, humidity, holiday and weekday. Finally, the Not Relevant category has the hour of the day, the wind speed and the working day.

2.5 Machine Learning Algorithms

There is a variety of Machine Learning (ML) methods that have been used for predicting bike-sharing behaviours. In this section the major ML algorithms considered in the state of the art references are described.

2.5.1 Random Forest model

Random Forest (RF) model is a supervised ML algorithm, which is an ensemble of several decision trees. Ensemble learning methods combine the decisions from multiple models which is expected to improve the overall prediction performance. RF can be applied both for classification and regression problems [28]. RF main advantages are:

- Fast training process and overall high accuracy;
- Can be easily implemented in parallel computational frameworks;
- Requires few hyper-parameters;
- Allows class prioritisation through class weight assignment.

RF is very successful particularly in multi-class object detection and in large-scale real-world computer vision problems. However, it has some disadvantages:

- The prediction is rather time-consuming;
- Requires significant computational resources.

The RF model is built on Decision Tree (DT) elements. Each DT has a number of nodes with different functionalities. The leaf node is a class label, determined by the majority vote of training examples reaching that leaf. A node is a question on one feature, for example "the temperature is higher or lower than a predefined threshold value ?". It branches out according to the answer. The root node is the first (top) node of the tree, the child node is the next node to a current node. Figure 2.10 shows an example of a

simple RF model with two DTs. Each DT has one root node with two child nodes. All other nodes have one parent node and either two child nodes or no child node. For any given input provided at the root node, the prediction is defined by the leaf nodes (red and green circles in Fig. 2.10). The RF model final prediction is the mean value of all DTs predictions.

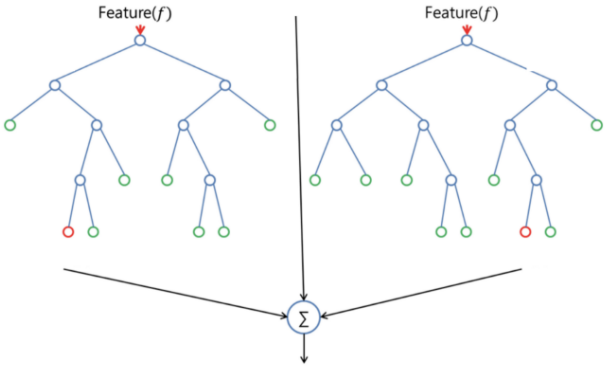


Figure 2.10: Simple Random Forest example [6]

The authors of [18] compare four methods, HA (Section 2.4.1), ARMA (Section 2.4.1), HP-MSI and P-TD and RF model, to predict the number of docked bikes at each station at a given time interval. It is demonstrated that RF outperforms the other methods. One possible reason is because RF model takes into account also meteorology data and time factors (holiday or not) in contrast to the other methods that only consider historical data of bike pickups/drop-offs.

2.5.2 Gradient boosting decision tree

Gradient Boosting Decision Tree (GBDT) is similar to RF algorithm (section 2.5.1). Both are ensemble learning methods (combining the decisions from all trees). The main difference, as described in [29], is the way the results are combined. The GBDT builds the decision trees sequentially is a way that each next tree is supposed to identify the errors from the previous trees and eventually correct them. In this way, GBDT often outperforms the RF. On the other hand, RF builds simultaneously and independently all trees. As a consequence, RF algorithm is much faster than GBDT.

Paper [19] compares the GBDT, the HA (Section 2.4.1), the ARMA (Section 2.4.1) and a hierarchical prediction based on a K-nearest neighbour (K-NN) algorithm. Two approaches are studied in the paper. In the first approach, the models try to predict the number of pickups across station clusters during all hours the system is open. In the second approach, only anomalous hours (i.e. only critical situations), are considered.

In both approaches, stations are clustered by an algorithm similar to K-means clustering (Section 2.5.6).

With respect to the first approach, the four studied algorithms (HA, ARMA, K-NN and GBDT) have similar errors in the range of 0.1. However, with respect to the second approach, HA and ARMA exhibit larger errors (in the range of 1-2), while GBDT and K-NN have errors between 0.6 and 0.8.

2.5.3 Neural Networks

Artificial Neural Networks (NNs) represent a variety of ML algorithms, inspired by biological nervous systems such as the brain. The NNs are particularly successfully in computer vision, natural language processing, bio-informatics, to mention a few. A standard (also known as shallow) NN consists of 3 layers (input, hidden and output layers). The input layer has as many nodes as are the features representing a particular dataset. The output layer produces the model predictions. An example of a shallow NN with three neurons in the input layer, five neurons in the hidden layer and two neurons in the output layer is represented in Figure 2.11. A sequence of hidden layers can build more complex functions of the network inputs and extract hidden representations of the raw data. Such complex NN architectures are known as deep learning (DL).

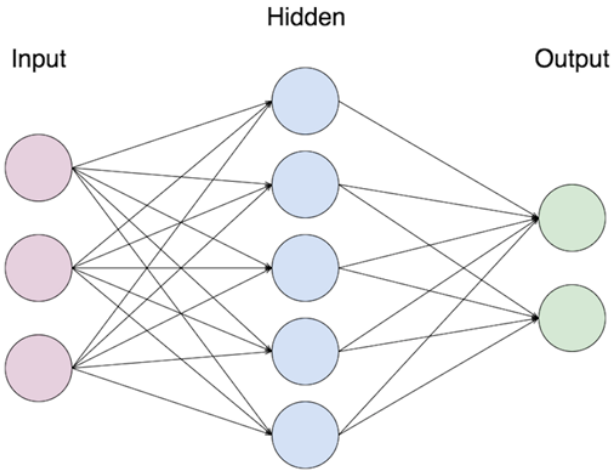


Figure 2.11: Shallow Neural Network architecture [7]

The NN is first trained with labelled training data. Its parameters are computed such that a predefined cost function is minimised. Once the parameters are optimised, the NN model is applied to new data for prediction purposes. Over the last few years, the NNs became also popular in the framework of BSS prediction. For example, in [24], a standard NN and a convolutional neural network (CNN) are comparatively studied combined with an ARMA model (Section 2.4.1). The results show that the CNN model outperforms the standard NN model.

CNNs lie at the core of the most powerful DL architectures working with images and video data, primarily due to their ability to extract representations that are robust to partial translation and deformation of input patterns. The key element of CNN is the convolution operation using small filter patches (kernels). These filters are able to automatically learn local patterns which can be combined together to form more complex features when stacking multiple CNN layers together. Within the stack of convolution layers, pooling layers are often placed intermittently. The pooling layers subsample the output of the convolution layers by outputting only one (e.g. max, mean) value for each small region. The subsampling allows the convolution layer after the pooling layer to work on a different scale than the layers before it. These features learned from the CNN are used as input to fully connected layers to perform classification, prediction, detection, segmentation.

The authors of [30] comparatively study the NN and GBDT models to transform the data into suitable inputs for a linear regression model in order to predict bike-sharing behaviours.

2.5.4 K-Nearest Neighbour

K-Nearest Neighbour (K-NN) is a classification technique usually applied when there is a little or no prior knowledge about the distribution of the data [31]. K-NN is based on the concept of distance (similarity) between records. To classify a new (unlabelled) record, K-NN computes its distance to all labelled records and identify K nearest neighbours (the k closest points to the new record). The class label of the new record is the label of the majority of the nearest neighbours. K is a design parameter that needs to be determined. Typical distance (similarity) measures applied in K-NN framework are the Euclidean Distance, Hamming Distance, Manhattan Distance and Minkowski Distance. K-NN can be described by the following steps:

1. Load the data.
2. Choose the number of the neighbours K.
3. For each new data entry (record):
   (a) Compute the distance between this entry and the entries from the given labelled dataset.
   (b) Add the distance and the index of the entry to a collection list;
4. Sort the distances and indices in increasing distance order.
5. Pick the first K entries from the sorted list.
6. In case of a regression problem, return the mean of the features of the selected K neighbours.

7. In case of a classification problem, return the label of the majority of the K nearest neighbours.

K-NN is a simple and easy to implement non-parametric algorithm, that does not go through a training stage. It is an instance-based learning model that evolves and adapts in the course of collecting new training examples. However, it has some disadvantages such as getting slow particularly when the labelled dataset is large. K-NN performs well when the number of features is relatively small, but suffers if the feature space has a high dimensionallity. The correct choice of the number of neighbours K is the main K-NN challenge. The search for the optimal K value requires running the algorithm several times which can be computationally very expensive.

2.5.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning model suitable for both classification and regression problems. In the classification framework, SVM is also known as a large margin binary classifier. The algorithm searches for a hyper-plane that provides the maximum margin between the two separated classes. This idea is illustrated in Figure 2.12. In the regression framework, SVM is known as Support Vector Regression (SVR).

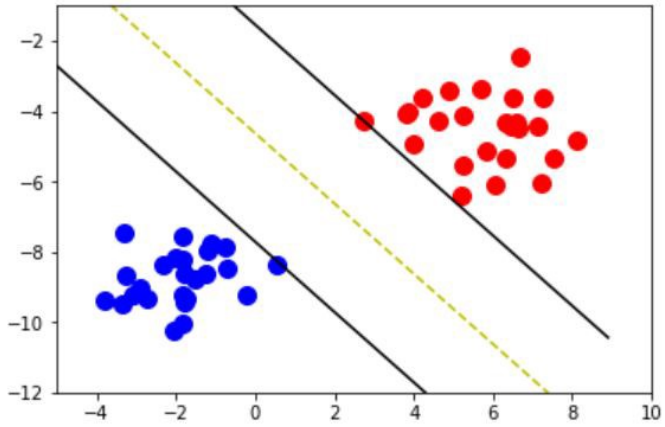


Figure 2.12: Two dimensional linear SVM [8]

The authors of [21] compare several regression models, *Ridge Regression*, *Ada-boost Regression* [32], *Support Vector Regression* [33], *Random Forest Regression* [28] and *Gradient Tree Boosting Regression* [34], to predict n hours ahead the number of bikes in use. Parameter n is a design parameter fixed as a number in the range of 1 to 24 hours. For each n, historical data from the past n hours before the time of prediction is used.

The results show that *Ridge Regression* and *Ada-boost Regression* are the best prediction models, particularly with respect to one hour prediction time. Although their prediction capacity deteriorates for longer prediction intervals, they still outperform the

other models. *Support Vector Regression*, *Random Forest Regression* and *Gradient Tree Boosting Regression* suffer of over-fitting problems and are not recommended as reliable solutions for the considered BSS scenario.

2.5.6 K-means Clustering

K-means clustering is an unsupervised ML algorithm (data is provided without labels) to automatically cluster similar data examples together. K-means is an iterative procedure that starts by guessing the initial cluster centers (aka centroids), and then refines this guess by repeatedly assigning data points to their closest centroids and then re-computing the centroids based on the assignments. The algorithm repeatedly carries out two steps: (i) assigning each training data point to its closest centroid, and (ii) re-computing the mean of each centroid using the points assigned to it. The K-means algorithm converges always to some final set of centroids. However, the converged solution may not always be ideal and depends on the initial setting of the centroids. Therefore, in practice the K-means algorithm is usually run a few times with different random initialisations. One way to choose the best clustering solution is to choose the one with the lowest cost function value (distortion). Figure 2.13 shows an example of a K-means algorithm result where each colour corresponds to a cluster.

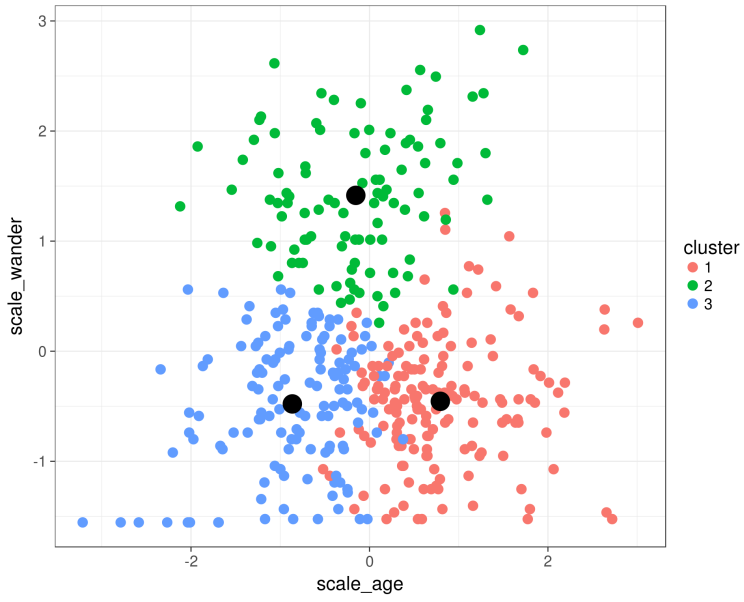


Figure 2.13: K-means clustering example [9]

Chapter 3

Decision Support System for Bewegen Bike-Sharing Systems

This chapter describes the development and implementation steps of the Decision Support System for Bewegen BSSs, which was conducted in three phases. The first phase, described in Section 3.1, is related with the collection of real data. The second phase, presented in Section 3.2, is related with data processing and visualisation. Finally, the third phase, described in Section 3.3, corresponds to the feature engineering approach to select the most representative features based on a ranking procedure. The selected features are used in next Chapter 4 to predict the bike-share traffic for the Bewegen BSS.

3.1 Data Collection

At the beginning of this project, the Bewegen BSSs had no means to collect in a systematic way the information on the usage of each station. Therefore, there was the need to develop and put in operation the appropriate mechanisms for data collection. Moreover, other types of data (weather and holiday information) had also to be collected from other external (i.e., Internet) services. Section 3.1.1 describes the developments done in the Bewegen backend system in order to collect station status data. Section 3.1.2 describes the service API used to collect information regarding the weather conditions. Section 3.1.3 describes the data collected regarding holidays in the city of the Begewen BSS used for collecting data.

3.1.1 Backend data collection module

In order to collect on-line, in an incremental manner, the historical data from all Bewegen systems, a new table was created in the backend database of each BSS named *System Status*.

Synchronous data collection approach. The first approach to feed this database

was to use the KPI (Key Performance Indicator) information available in the Bewegen systems. If this would be possible, the implementation of data collection module would be relatively simple and the extra computational effort for Bewegen servers would be minimal. Bewegen systems use a set of KPIs which are updated every 2 hours. In fact, the updated values of some of these KPIs are provided on the system webpage (number of active members, total number of trips, favourite stations for pickups or drop-offs, etc.) but the ones of interest were only active for debugging purposes. The KPIs of interest to this dissertation are the number of parked bikes and the number of free docks on each station. Since these KPIs were already available, the idea of the first approach was to make their update more frequent (e.g. every 15 minutes) and start storing all updated values in the *System Status* table together with their timestamp values. One question that had to be considered in this approach was how frequent these updates should be done.

After consulting several references on BSS prediction solutions, it became clear that this approach has a lot of problems. First, since we do not have the information on the pickups and drop-offs between two consecutive updates, the occupation information can be inaccurate. For example, on a given time period, if a sequence of three drop-offs occurs and is followed by a sequence of three pickups on a given station, the average occupation at this station will be different from the value registered in the KPI associated with the number of parked bikes. Second, because there is no information on each pickup or drop-off, it is also not possible to know which type of lock is used when a bike is parked. As explained in Section 2.1, there are two different ways to lock a bike: clients can either park the bike in one of the available docks (primary lock), or attach the bike to another bike using its padlock (secondary lock). Although there are no rules, typically, clients use primary lock unless all the docks are occupied. Since it was important to know what action type (pickup or drop-off) and lock type (primary or secondary) occurs, this approach was discarded.

Asynchronous data collection approach. Instead of periodically collecting the status of each station, the collecting operation is triggered every time an action (pickup or drop-off) occurs. In this way, instead of storing the system status information in fixed periods of time, this approach is asynchronous: when an action occurs, a new row is added to the *System Status* table with the information of the action. Following this approach, all relevant data from each station is collected.

Note that, by default, the bikes send a message to the server when a pickup/drop-off occurs, and the server handles this message by generating events which afterwords are sent to a message broker to be processed. Based on this characteristic of the system, a module was created that whenever one of these messages arrive to the message broker, the system checks the status of the station related to this event and stores all useful information in the *System Status* table.

The *System Status* table was implemented with each row representing an action and the different columns containing all relevant information collected when the action occurs. Each row has 8 columns, with the following information:

1. Tenant: a string with Bewegen system's name where the action occurred (String)

2. Id: unique ID of the action (UUID)
3. StationId: unique ID of the station where the action occurred (UUID)
4. StartAt: the UTC time instant when the action occurred (Sec)
5. FreeSpaces: number of available docks in the station after the action (Integer)
6. OccupiedSpaces: number of occupied docks in the station after the action (Integer)
7. SecondaryLockFreeSpaces: number of available Secondary Locks in the station after the action (Integer)
8. SecondaryLockOccupiedSpaces: number of used Secondary Locks in the station after the action (Integer)

Note that, there is no explicit columns to indicate the action type or the associated lock type. This fact is because this table was created for multiple purposes and not only for the purpose of this dissertation. The action type and associated lock type were computed from the table information in the following way:

- The associated lock is calculated by comparing which column value changes between this action row and the previous row of the same station. If the OccupiedSpaces column has different values in both rows, it means that it is a primary lock. Otherwise, if the OccupiedSpaces column has the same value in both rows and SecondaryLock-OccupiedSpaces column has different values, it means that it is a secondary lock.
- The action type is calculated taking into account the associated lock. If it is a primary lock, the action type is calculated by subtracting the number of OccupiedSpaces from this row to the number of OccupiedSpaces from the previous row (of the same station). If the result is one, it means that the number of occupied docks decreased and, therefore, this action is a pickup. On the other hand, if the result is minus one, it means that there is one less occupied dock and, so, this action is a drop-off. A similar calculation is done if the associated lock type is of secondary type. The action type is calculated by subtracting the number of SecondaryLockOccupiedSpaces of this row to the number of SecondaryLockOccupiedSpaces of the previous row. If the result is one, it means that the number of secondary locks being used decreased and this action is a pickup. On the other hand, if the result is minus one, it means that there is one less secondary lock being used and this action is a drop-off.

After analysing the first month of data collected in this table, a problem was identified. As explained before, every time that an action occurs, a message is sent to a message broker to be processed. In normal operation, there is no queuing of messages in the broker. Nevertheless, in the case when multiple actions occur in a small time interval in the same station (i.e., very close to each other), the message broker has a queue where messages are

stored while they are waiting to be processed. In this case, sometimes a problem occurs where the values of the received messages are not properly handled and, as a result, the generated events can swap values between actions.

To solve this problem, a more reliable message broker need to be developed in the near future (this development was outside the scope of this dissertation). Moreover, as already referred, this problem occurs only when the multiple actions occur in the same station, which happens rarely.

Nevertheless, to eliminate data collected with errors, the following processing was also implemented. In both lock types (i.e., primary lock and secondary lock), if the subtraction does not produce a value of one or minus one, the action is discarded. Moreover, if both OccupiedSpaces and SecondaryLockOccupiedSpaces values are simultaneously different or simultaneously equal, it means that there was an error processing the data of this action and this action is discarded as well.

3.1.2 Weather data collection

Another important factor in order to predict the occupation of system stations is the weather conditions. It is easily foreseen that factors such as temperature and rain deeply impact the usage of BSSs. Moreover, there are even BSSs that close on seasons that have more extreme weather conditions. As an example, in Bewegen systems, half of the ones in United States of America close in winter.

There are many reliable web services for weather information. Services like AccuWeather and YahooWeather were considered but at the end it was decided to use OpenWeatherMap for collecting weather data. In every weather request in a city of choice, the OpenWeatherMap Api returns information about the current weather in the time instant of the request. The response contains the following fields:

1. Weather description (UUID)
2. Temperature:
   (a) Actual Temperature (Double)
   (b) Maximum Temperature (Double)
   (c) Minimum Temperature (Double)
3. Atmosphere:
   (a) Pressure (Double)
   (b) Humidity (Double)
4. Wind:
   (a) Speed (Double)
   (b) Degree (Double)

5. Rain (Double)

6. Snow (Double)

In order to collect the proper data from OpenWeatherMap Api, the first approach was to create a cron job (a job that executes a given set of scripts at a given predefined time instant) which would run three times a day. The idea was to run this job executing a python script (which calls the OpenWeatherMap Api) and store the retrieved information in a Postgresql database. Initially, collecting weather data three times a day seemed enough to obtain the needed data for the purpose of this dissertation. This solution was implemented and run for two weeks.

Nevertheless, after better analysis of the OpenWeatherMap Api documentation, another solution was used to collect the required weather information. OpenWeatherMap has a historical service which provides the weather conditions for all hours of all days between two dates and in a given city (the city, the starting date and the ending date are the request parameters). This service has an associated cost but, after discussing this issue with Bewegen and since the cost was not too high, it was decided to use this historical service instead of the first approach.

3.1.3 Holidays data

Another important factor for BSS prediction is the information about holidays or special events. This factor affects in a different way, the various types of clients. On one hand, the usage of bike-sharing in holidays is expected to rise since normally more people have the day off. On the other hand, the usage might decrease among clients using BSS between their home and working places. In both cases, it is expected changes in the normal usage of bikes. The information about the holidays was taken from the website *http://www.webcal.fi*. For each holiday, the information given by this website is the following:

1. Date (Sec)

2. Name (String)

3. Alternative name (only if exists) (String)

4. Description (String)

To collect the holiday information, it is only needed to go to the website and download a file with all the information. In this case, this service is free.

3.2 Data Visualisation and Processing

This section describes the data visualisation and pre-processing conducted in order to prepare the data for the prediction of machine learning (ML) algorithms.

3.2.1 Data Visualisation (Station Occupation)

Important to refer that data collection started at the beginning of January 2019. Therefore, the first data visualisations were based on less data. As a consequence, it was observed that the number of Secondary Locks was usually very close to zero in all stations. Therefore, it was decided to consider the occupation as a ratio between the number of docks occupied by bikes (i.e. the number of Primary Locks) and the total number of available docks at the station.

In order to have a better understanding of the Bewegen system usage dynamics, the collected data for each station was visualised to check the existence of particular patterns on the station status as time goes on. The occupation (in percentage) observed in January in four representative stations is shown in Figures 3.1 to 3.4.

The following conclusions can be made from the figures. The two stations with more traffic (more total number of pickups and drop-offs) were *Station 1* (Figure 3.1) and *Station 2* (Figure 3.2). In January, *Station 3* was the only one that reached 0% occupation (i.e., no bikes available at the station) and it happened only in one day (observed in Figure 3.3). Although such situation is normally rare in the Winter season, it is exactly the type of situations the Decision Support Service needs to prevent. On the other hand, no station has reached its full capacity (all available docks at the station to be occupied), with the maximum observed occupation around 81% in *Station 11* (Figure 3.4) and almost 80% in *Station 2* (Figure 3.2). Finally, *Station 11* is, among all stations, the one that presents the highest range of occupation, varying during this month from 3% up to 81%.

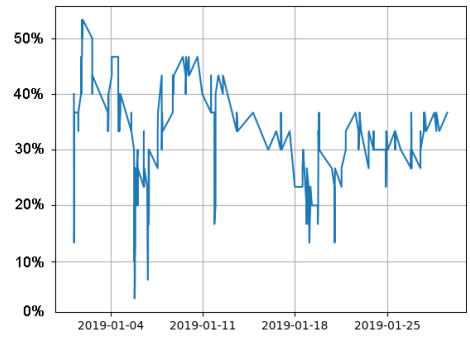


Figure 3.1: *Station 1* occupation percentage in January

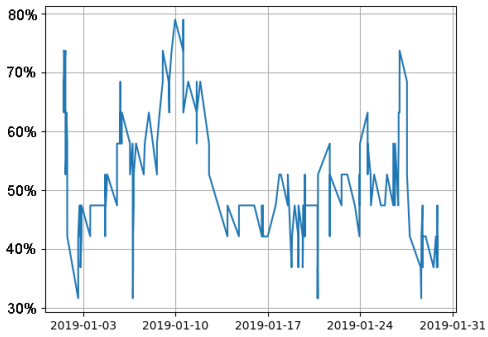


Figure 3.2: *Station 2* occupation percentage in January

3.2.2 Feature Matrix Construction - First Version

In the previous section, data visualisation was based on the timestamp of each action (pickup or drop-off). The next logical step was to divide the days into equal periods of time and see how many pickups and drop-offs have occurred in each interval. To this aim,

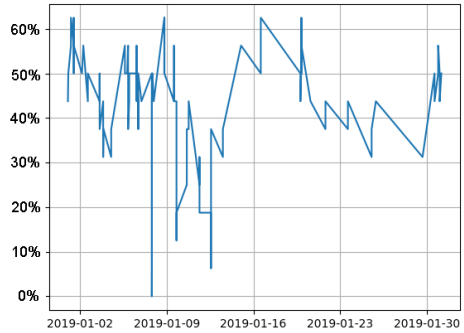


Figure 3.3: *Station 3* occupation percentage in January

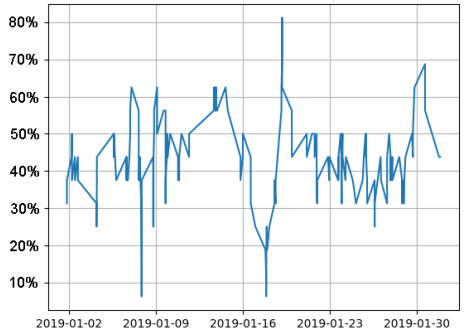


Figure 3.4: *Station 11* occupation percentage in January

a python script was developed in order to process the *System Status* information. The script runs the following tasks.

Task 1 The script divides the input data (*System Status* information) into one hour time intervals and counts the number of pickups and drop-offs occurred on each time interval. The 1 hour time interval was chosen because the weather API also returns hourly weather updates.

Task 2 The script converts the timestamp of each action into the local hour of the system location (this is necessary because the timestamps collected by Bewegen servers are always defined in UTC time zone).

Task 3 The script processes the weather information considering only 3 weather indicators: the temperature, the rain and the snow.

Concerning the temperature, since it can have a wide range of values, and to facilitate the ML prediction algorithms, instead of using the temperature absolute value, it was decided to cluster the values into three categories: Cold, Medium and Hot. The range limits between the three categories for a given time period are based on the minimum temperature t_m and the maximum temperature t_M of all historical data on the same time period. Then, the range limits are calculated assuming equal temperature intervals for all categories, i.e., the range limit between Cold and Medium is:

$$t_m + \frac{(t_M - t_m)}{3}$$

and the range limit between Medium and Hot is:

$$t_m + \frac{2 \times (t_M - t_m)}{3}$$

Concerning the rain and the snow, these two weather indicators are aggregated on a single binary feature which is equal to 1 if it is either snowing or raining, or equal to 0 otherwise (i.e., it is neither snowing nor raining).

Task 4 The script places all data (either collected or processed) into a table (which is saved in a csv file) where each row has all information related to each 1 hour time period. Each row of the table has the following fields:

1. Day: day of the month of the time period (Integer)
2. Month: month of the year of the time period (Integer)
3. Year: year of the time period (Integer)
4. Hour: initial hour of the time period (Integer)
5. Pickups: number of pickups occurred in the time period (Integer)
6. Drop-offs: number of drop-offs occurred in the time period (Integer)
7. Day of week: day of week of the time period represented by an integer value from 0 (Monday) to 6 (Sunday) (Integer)
8. Is_holiday: value indicating if the day of the time period is a holiday (Binary)
9. Temperature: a string indicating the temperature category in the time period (String)
10. Rain/Snow: value indicating if it is raining or snowing in the time period (Binary)

3.2.3 Data Visualisation (Pickups)

This data visualisation phase was conducted when 7 weeks of collected data were available and, so, the data visualisation results of this section are based on this dataset.

The first step of data visualisation phase was to observe the number of pickups in the whole system. Figure 3.5 shows the minimum, average and maximum number of pickups per hour of the day among all days and the same values among the different days of each week. The main goal was to understand if there were different behaviours at different days of week and at different hours of the day. As observed in Figure 3.5, data seem uniformly distributes in all cases, which means that the day of week and the hour of the day in the winter season do not have a major influence in the system usage.

The next step of this phase was to observe the number of pickups on a single station. *Station 1* was selected as it is the station with more usage in terms of number of pickups and drop-offs. Figures 3.6 to 3.12 show the minimum, average and maximum number of pickups per hour of the day among the different days of a week.

By analysing these figures, one can reach the following observations. The maximum number of pickups per hour is two but most of the hour periods have at most one pickup. In most of hour periods, there are no pickups the reason why the average number of pickups per hour is less than 0.2.

With these observations, we can conclude that the number of pickups is well spread over the 24 period of time. This is unexpected because normal behaviour is to have more

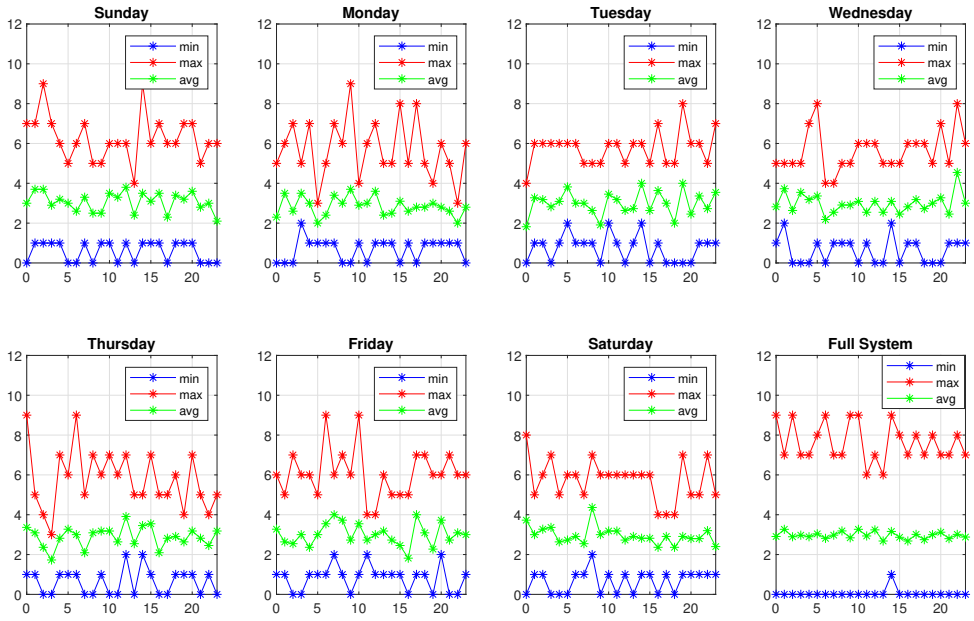


Figure 3.5: Number of pickups per day of week

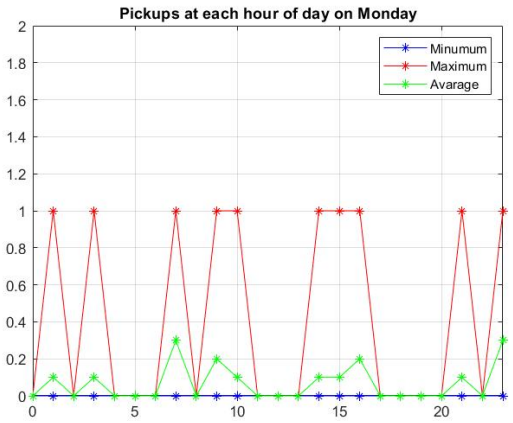


Figure 3.6: Pickups on Monday in *Station 1*

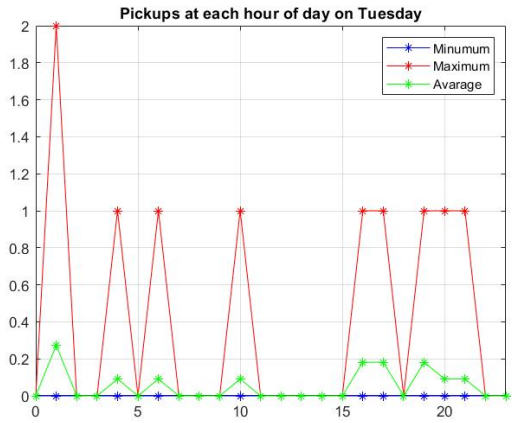


Figure 3.7: Pickups on Tuesday in *Station 1*

pickups at day time. One possible reason for this behaviour is the low system usage of the observed data which corresponds to winter time.

Moreover, clients are using the system more in weekdays than in weekends. Usually this behaviour happens in BSSs where there are more clients using the system in day-to-day activities than clients visiting the city.

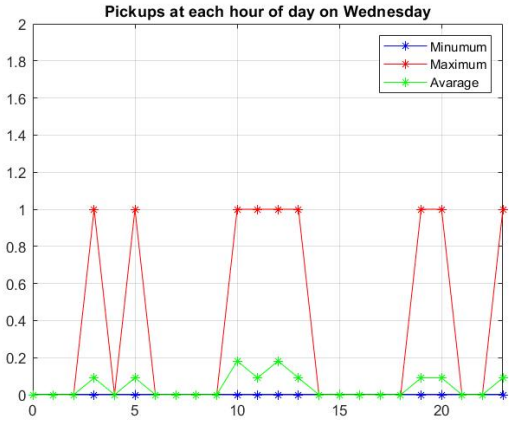


Figure 3.8: Pickups on Wednesday in *Station 1*

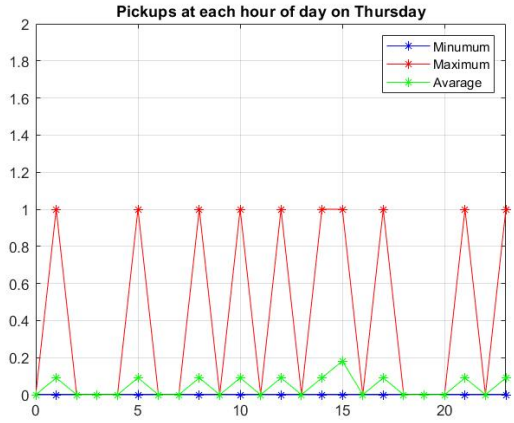


Figure 3.9: Pickups on Thursday in *Station 1*

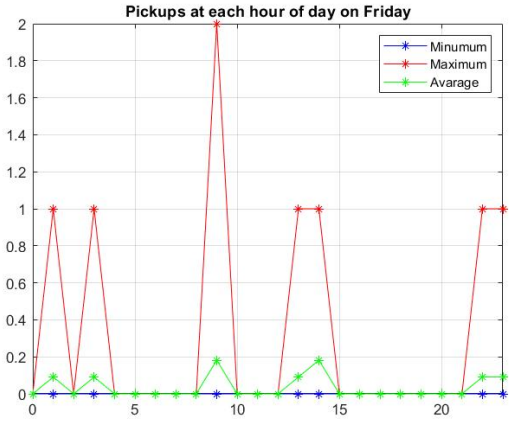


Figure 3.10: Pickups on Friday in *Station 1*

Figure 3.11: Pickups on Saturday in *Station 1*

3.2.4 Data Visualization (Weather)

Another important feature that impacts the usage of BSSs is the weather. As explained before, the temperature feature is categorical and can be Cold, Medium or Hot. Figure 3.13 shows the impact on the number of pickups on each day of week the system is in each of the temperature categories.

Much like the previous observations, not much can be concluded, mainly due to the low system usage of the observed data. In fact, it was expected that warm weather would increase the number of pickups which is not always the case. On Mondays, Tuesdays and Wednesdays, the average number of pickups is higher with Hot temperatures but for all the other days of week, that is not the case. Moreover, in all cases, the difference in the number of pickups between the 3 temperature categories is small.

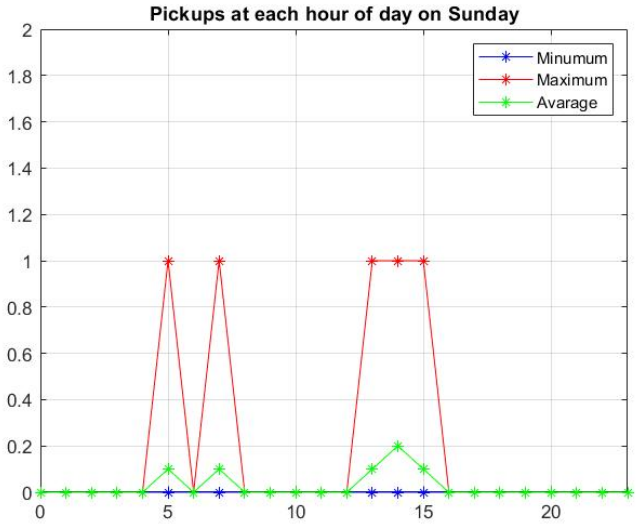


Figure 3.12: Pickups on Sunday in *Station 1*

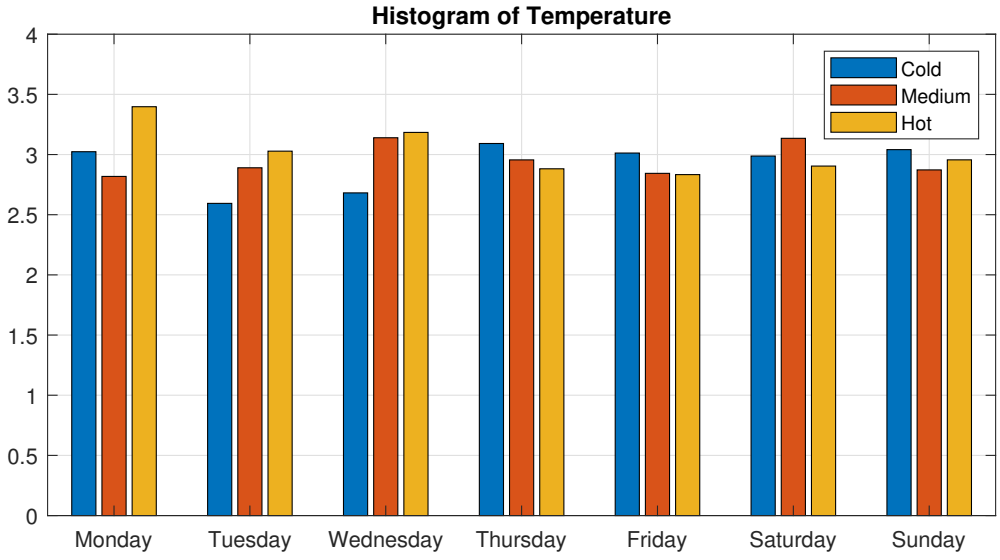


Figure 3.13: Average number of pickups for the 3 temperature categories at each day of week

3.2.5 Feature Matrix Construction - Final Version

As concluded before, the number of pickups (and drop-offs) in the system stations is not high and this was mainly due to the fact that collected data happened on winter. It is known that prediction methods do not perform well with a lot of zeros as input data. Since the occupation is almost never zero in any station and predicting the stations' occupation

(in number of bikes) is actually the ultimate goal of this dissertation, a new approach was adopted where instead of studying the number of pickups, the main focus was changed to the occupation of the stations.

A few number of changes had to be done in the previous python script in order to cope with this change. First, instead of dividing the data into 1 hour time intervals, it was decided to change it to 30 minute time intervals, which was the initial goal for the short term prediction task (Section 1.2). Second, instead of the number of pickups and drop-offs, the script has to calculate the average occupation on each time interval. The average occupation is calculated by adding the occupation given by the OccupiedSpaces column (see Section 3.1.1) of the *System Status* table associated with all actions occurred in the time interval divided by the number of such actions. Third, it was decided to discard the night hours (from slot 23:00-23:30 until slot 8:30-9:00) of the system because there was not enough traffic in stations to be considered relevant for the prediction algorithms. Therefore, in the resulting datasets, each day has a total of 28 time intervals (each 30 minutes over 14 hours)

3.3 Description of Features

This section describes all selected features to be used as input to the ML algorithms in Chapter 4 and how they were ranked in terms of their importance.

3.3.1 Features

All state of the art approaches use the historical values in order to predict bike-sharing usage. The difference between different approaches is on which values are used. In this dissertation, it was decided to include all the occupation historical data from the 9:00-9:30 to 22:30-23:00.

Two discrete features are considered: the Hour feature which indicates for each time interval, the hour of day of its initial time instant and whose values are all integers from 0 to 23; and the Day_of_week feature whose values are from 0 (representing Monday) to 6 (representing Sunday).

Moreover, some binary features are also considered. First, the feature Is_holiday which indicates if the day of the current time period is holiday or not. Second, the feature Rain/Snow, which indicates if the day of the current time period is raining/snowing or not. Third, the feature Is_today, a runtime feature which is used to give more importance to the historical occupation values of the same day as the current time period (this feature is only used in the K-nearest neighbour algorithm). As already stated, each time interval takes 30 minutes and, so, the feature Minutes is binary with the possible values being 0 (representing 0 minutes) and 1 (representing 30 minutes). Like the Hour feature, the Minutes feature represents the initial time instant of the time interval. For example, the interval 10:00-10:30 has the associated features Hour = 10 and Minutes = 0, while interval 10:30-11:00 has the associated features Hour = 10 and Minutes = 1.

The categorical features that were considered are the Temperature and the Month of the year. In order to encode the values of these features as input to the ML algorithms, it was used the One Hot Encoding.

One Hot Encoding is a method by which categorical feature values are converted into a more appropriate form for ML algorithms (this method avoids the categorical values to have different levels of importance between them). The method consists in creating a binary feature for each possible value of the categorical feature. Then, each binary feature is 1 when the categorical feature has its associated value. In the case of temperature, the method turns the categorical values Cold, Medium and Hot into three binary features T_Cold, T_Medium and T_Hot as illustrated in Table 3.1.

Temperature	T_Cold	T_Medium	T_Hot
Cold	1	0	0
Medium	0	1	0
Hot	0	0	1

Table 3.1: One Hot Encoding of Temperature feature encoded in 3 binary features T_Cold, T_Medium and T_Hot

The One Hot Encoding method is also used for Month of the year. In the current implementation, since the dataset of this dissertation is from January to April, the considered number of binary features is four (as illustrated in Table 3.2). Nevertheless, the system was implemented in a way such that as soon as more months appear as values of the Month of the year feature, the One Hot Encoding automatically extends the number of binary features.

Month	M_January	M_February	M_March	M_April
January	1	0	0	0
February	0	1	0	0
March	0	0	1	0
April	0	0	0	1

Table 3.2: One Hot Encoding of Month of year in a binary feature per month

Note that by using One Hot Encoding, the matrix with the input features becomes larger which might impose in general extra computational costs. In the case of this dissertation, this was not a problem since the resulting number of features is not that large.

3.3.2 Ranking features

Section 3.3.1 described all features taken into account in this dissertation. In order to improve ML predictions, it is also important to rank the features according to their

influence on the data. For feature ranking, it was used a python library called *Sklearn*, one of the most used ML libraries for python.

The features were ranked based on the *SelectKBest* class of *Sklearn* library. This class scores the features according to an input scoring function. Since in this dissertation we are dealing with float values (representing average occupation values) there are only two possible scoring functions available: *f_regression* and *mutual_info_regression*. It was decided to use function *f_regression*, a linear model that calculates the influence of each feature regarding all features. This model was applied to the following features:

- Hour (Hour of the day between 9 and 23)
- Minutes (binary value - 0 for 0 minutes and 1 for 30 minutes)
- Temperature, encoded as the 3 features T_Cold (binary value), T_Medium (binary value), T_Hot (binary value)
- Day_of_week (value between 0 and 6)
- Is_holiday (binary value)
- Rain/Snow (binary value)
- Month of the year, encoded as the 4 features M_January (binary value), M_February (binary value), M_March (binary value), M_April (binary value)

The ranking results of applying *f_regression* to these features are shown in Table 3.3. Curiously, the features related with the type of days (either the Day_of_week and the Is_holiday) are the most influential features on the collected data while the features related with the time intervals on each day (Hour and Minutes) are the least influential. Moreover, among the Temperature features, the T_Hot and T_Medium features are much more important than the T_Cold feature.

Rank	Features	Ranking
1	Day_of_week	3.0452e+01
2	Is_holiday	2.3022e+01
3	T_Hot	9.5574e+00
4	T_Medium	8.8768e+00
5	M_January	5.0890e+00
6	M_March	3.4212e+00
7	Rain/Snow	4.7335e-01
8	T_Cold	3.0409e-01
9	M_February	2.6692e-01
10	Hour	2.5395e-01
11	M_April	1.3940e-02
12	Minutes	7.6795e-03

Table 3.3: Scores and ranking of each feature with *f_regression*

Chapter 4

Prediction of Bike-Share Usage with Machine Learning

This chapter describes the ML algorithms proposed in this dissertation for predicting the number of bikes in each station. Section 4.1 describes the dataset. Section 4.2 describes how the test error is calculated in the prediction algorithms. Section 4.3 describes the studies done with the Neural Network (NN) prediction model. Section 4.4 describes the studies done with the k-Nearest Neighbour (K-NN) model. Finally, Section 4.5 draws the main conclusions.

4.1 Dataset Description

Since the aim is to predict the number of bikes on each BSS station, each station has its own individual dataset. In order to train and test the ML models, 16 stations (generically named from *Station 1* to *Station 16*) from one of Bewegen's systems were selected.

As already mentioned, the data collecting process was firstly developed and put in operation and, then, data started being collected. The final dataset used in this work is the historical data collected from 1 of January 2019 until 30 of April 2019. Since time is divided in 30 minute time intervals, each station dataset has 3324 rows of data (containing all the feature values, as described in Section 3.3.1).

4.2 Error Calculation

In order to evaluate the implemented ML models, an error metric needs to be defined. The average of the absolute values of the differences between the predicted values and the real values of all predicted cases is typically used in the relevant literature. This error is computed by the following formula where c is the number of predicted cases.

$$error = \frac{1}{c} \sum_{i=1}^{c} \mid predicted\_value_i - real\_value_i \mid \tag{4.1}$$

Recall that the historical data values are the average number of bikes in a station for every 30 minute time interval and, so, the results are float values. This means that regression is the relevant ML approach to be applied to this problem. Nevertheless, the final prediction needs to be an integer value because in a business standpoint, it is more meaningful to provide an integer value indicating an estimated number of bikes.

Consider a case of a client searching for a nearby station with available bikes in the next 30 minutes. In this case, there is no problem if there are more bikes in a station than the predicted number of bikes. On the other hand, if the prediction is that some bikes are in station and, then, no bikes are available, the client satisfaction on the service will degrade drastically. Note that the other extreme case is not a problem in Bewegen systems since when the station is full (no available docks), the clients can always lock the bikes through the padlock as Secondary Locks (see Section 2.2).

Since it is preferable that the estimated value is lower than the real value (i.e., it predicts a lower number of bikes), the final estimated value is computed as the regressor result rounded down, i.e., the highest integer value not higher than the regressor estimated value.

4.3 Neural Network

The choice of using an artificial NN in this dissertation was mainly because some state of art articles report good accuracy results of this algorithm when applied to bike-sharing estimation tasks (see Section 2.5.3). A standard 3 layers (input, hidden, output) NN was implemented by using the *MLPRegressor* class from the python library *Sklearn*, the same library used for ranking system features (Section 3.3.2).

In the initial step, the NN was tested with the dataset associated to the station with more traffic in terms of total number of pickups and drop-offs, named *Station 1* (see Section 3.2.1). As input values for the NN model, in addition to the features described in Section 3.3.2, the historical values of the last 27 time intervals were also included. So, there are 39 input features which are the following ones:

1. Day_of_week (Integer from 0 to 6) - 1 feature
2. Is_holiday (Binary) - 1 feature
3. T_Cold (Binary) - 1 feature
4. T_Medium (Binary) - 1 feature
5. T_Hot (Binary) - 1 feature
6. Rain/Snow (Binary) - 1 feature
7. Hour (Integer from 0 to 23) - 1 feature
8. Minutes (Binary) - 1 feature

9. M_January (Binary) - 1 feature
10. M_February (Binary) - 1 feature
11. M_March (Binary) - 1 feature
12. M_April (Binary) - 1 feature
13. Historical data: occupation values of the previous known 27 time intervals (Float) - 27 features

The NN has only one output value which is the estimated average number of bikes on a given station for a given time interval. For example, if the current time is between 16:00 and 16:30, the more recent historical time interval is 15:30-16:00. Then, in the short term prediction task, the aim is to predict the average number of bikes that will happen during the current time interval. In the long term prediction task, the aim is to predict the average number of bikes between 16:00 and 16:30 of the next day.

So, the time interval to be estimated is: (i) the current time interval in the short term prediction task or (ii) the time interval 24 hours after the current one in the long term prediction task.

In order to get the best possible prediction accuracy, two NN parameters have to be selected: the number of neurons in the hidden layer (n) and the regularisation value (α) which helps avoiding overfitting (by penalising for high feature weights). Moreover, the input features that provide the best accuracy results have also to be selected. Recall that the features are already ranked (see Section 3.3.2) and the question is which of the most ranked features should be considered.

For testing the model, the full dataset of *Station 1* was divided in two datasets: the training dataset which has 60% of the four months historical data and the test dataset with the remaining 40%. Concerning the number of neurons (n), the values from 2 to 20 neurons in the hidden layer were tested. Concerning the regularisation value α, six values were considered: 0.0001, 0.001, 0.01, 0.1, 0.3 and 0.5.

Next, the NN model selection results (i.e., the selection of the number of neurons n and regularisation value α) are described separately for the short term prediction task (Section 4.3.1) and for the long term prediction task (Section 4.3.2). Then, the feature selection results are described in Section 4.3.3 for both tasks. Finally, Section 4.3.4 reports the best parameters and resulting average test errors applied in all other 15 stations.

4.3.1 NN model selection for *Station 1* - short term prediction

Table 4.1 presents the test errors of the NN algorithm for all combinations of n and α. In order to better understand the results, Figure 4.1 plots the test error values obtained by each α (alpha in the legend of the figure) as a function of the number of neurons n.

The results show that the minimum error is 1.003 (highlighted in Table 4.1) which is obtained by having 9 neurons in the hidden layer and a regularisation value of 0.5.

	α =0.0001	α =0.001	α =0.01	α =0.1	α =0.3	α =0.5
$n = 2$	1.009	1.314	1.020	1.343	1.077	1.106
$n = 3$	1.084	1.035	1.254	2.170	1.115	1.080
$n = 4$	1.071	1.068	1.060	1.007	1.157	1.028
$n = 5$	1.039	1.026	1.084	1.079	1.045	1.076
$n = 6$	2.052	1.071	1.048	1.071	1.019	1.062
$n = 7$	1.033	1.040	1.066	1.030	1.128	1.174
$n = 8$	1.099	1.046	1.192	1.033	1.084	1.011
$n = 9$	1.111	1.019	1.064	1.086	1.118	**1.003**
$n = 10$	1.017	1.017	1.056	1.070	1.021	1.063
$n = 11$	1.196	1.164	1.024	1.018	1.055	1.235
$n = 12$	1.191	1.124	1.136	1.105	1.088	1.129
$n = 13$	1.103	1.074	1.049	1.022	1.066	1.058
$n = 14$	1.025	1.161	1.108	1.046	1.030	1.094
$n = 15$	1.108	1.135	1.019	1.055	1.088	1.043
$n = 16$	1.129	1.072	1.132	1.060	1.014	1.030
$n = 17$	1.074	1.015	1.043	1.080	1.014	1.027
$n = 18$	1.029	1.071	1.040	1.057	1.067	1.013
$n = 19$	1.111	1.169	1.127	1.022	1.008	1.037
$n = 20$	1.019	1.021	1.075	1.119	1.013	1.137

Table 4.1: NN Model selection (number of neurons and alpha parameter) on short term prediction

Nevertheless, there are many other combinations of values providing a test error value very close to the minimum one, showing that the NN accuracy is not much penalised if the adopted parameters are not the optimal ones. This observation is easily seen in Figure 4.1 where most of the cases have an error value below 1.2 and only two cases have a really bad result.

4.3.2 NN model selection for *Station 1* - long term prediction

Similarly to the previous section, Table 4.2 presents the test error value of the NN algorithm for all combinations of n and α. Again, in order to better understand the results, Figure 4.2 plots the test error values obtained by each α as a function of the number of neurons n.

In this case, the results show that the minimum error is 1.011 (highlighted in Table 4.2) which is obtained by having 12 neurons in the hidden layer and a regularisation value of 0.0001. Once again, the NN accuracy is not much penalised if the adopted parameters are not the optimal ones since there are many other combinations of values providing a test error value very close to the minimum one. This observation is again easily seen in Figure

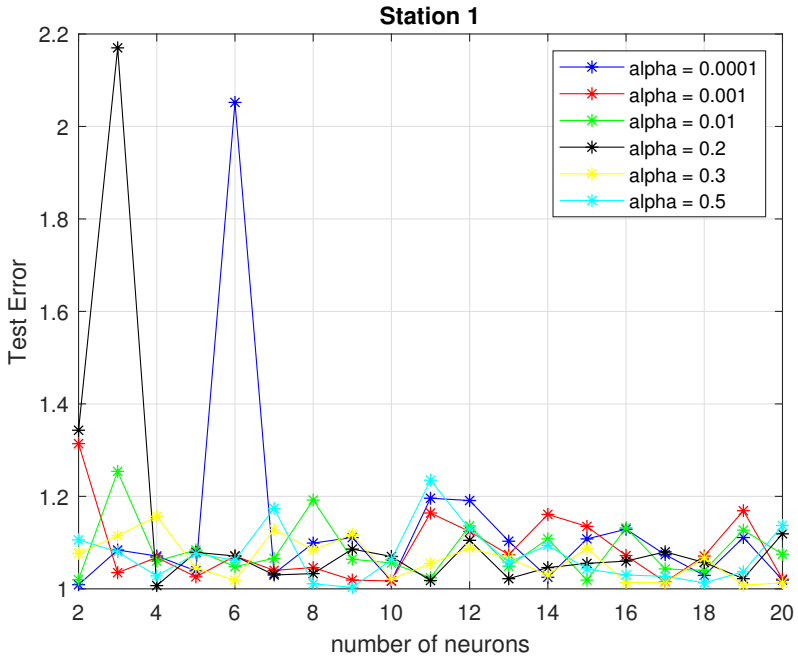


Figure 4.1: Test error as a function of number of neurons (n) and regularisation param. (α) on short term prediction

4.2 where only four cases have a really bad result.

4.3.3 Features selection in NN-based prediction

After the search of the optimal values for the number of neurons in the hidden layer and the regularisation parameter, the next step is to study if it is possible to further reduce the test error by removing the least important features (defined by the rank reported in Section 3.3.2). In this step, the number of most important features is defined as p and the tested values of p are all possible ones, i.e., from 1 (only the most important feature) up to 39 (all features). For each value of p, the NN model was run ten times and the result is the mean test error value among all 10 values.

As determined before, the best NN model for short term prediction task is when the value of neurons in hidden layers is 9 and the regularisation parameter is 0.5. For long term prediction task the best NN model is when the number of neurons in hidden layer is 12 and regularisation parameter is 0.0001. Figure 4.3 shows the average test error obtained by these two models as a function of the selected number of most important features p.

The results in Figure 4.3 show that the best test error has improved from the previous 1.003 to 0.978 for the NN short term prediction task by using the 12 best ranked features. Similarly, the results show that the best test error has improved from the previous 1.011 to 0.987 for the NN long term prediction task by using the 6 best ranked features. So, the main conclusion is that indeed reducing the input number of features improves the accuracy

	α =0.0001	α =0.001	α =0.01	α =0.1	α =0.3	α =0.5
$n = 2$	1.925	1.125	1.077	1.101	1.098	1.926
$n = 3$	1.925	1.229	1.926	1.135	1.113	1.058
$n = 4$	1.057	1.087	1.047	1.170	1.017	1.040
$n = 5$	1.117	1.096	1.041	1.121	1.106	1.033
$n = 6$	1.031	1.116	1.046	1.190	1.029	1.119
$n = 7$	1.033	1.106	1.091	1.117	1.054	1.044
$n = 8$	1.077	1.171	1.070	1.174	1.090	1.106
$n = 9$	1.217	1.084	1.076	1.113	1.071	1.114
$n = 10$	1.064	1.173	1.149	1.092	1.085	1.084
$n = 11$	1.041	1.049	1.099	1.029	1.046	1.145
$n = 12$	**1.011**	1.041	1.130	1.079	1.122	1.092
$n = 13$	1.052	1.044	1.180	1.051	1.046	1.048
$n = 14$	1.044	1.182	1.195	1.063	1.094	1.210
$n = 15$	1.046	1.092	1.121	1.175	1.090	1.128
$n = 16$	1.051	1.175	1.162	1.105	1.059	1.089
$n = 17$	1.068	1.071	1.111	1.050	1.087	1.071
$n = 18$	1.116	1.055	1.137	1.026	1.045	1.090
$n = 19$	1.165	1.085	1.057	1.054	1.103	1.044
$n = 20$	1.038	1.048	1.139	1.062	1.059	1.045

Table 4.2: NN Model selection (number of neurons and alpha parameter) on long term prediction

of both NN models while also reducing the computational costs of their execution.

4.3.4 Results for the Other Stations

The study previously described for *Station 1* was then repeated for all other 15 stations. Table 4.3 presents the n, α and p values that have provided the best test error and the obtained test error value for all other stations in the short term prediction task, while Table 4.4 provides the same information for all other stations in the long term prediction task. The analysis of these results will be done later in Section 4.5.

4.4 K-Nearest Neighbour

The second algorithm used in this dissertation is K-NN, which is mentioned in some works of the state of the art. This method was implemented by using the *KNeighborsRegressor* class from the python library *Sklearn*. Like in the previous ML method, the K-NN was first tested with the dataset associated to *Station 1*. Recall that the time interval to be estimated is: (i) the current time interval in the short term prediction task or (ii) the

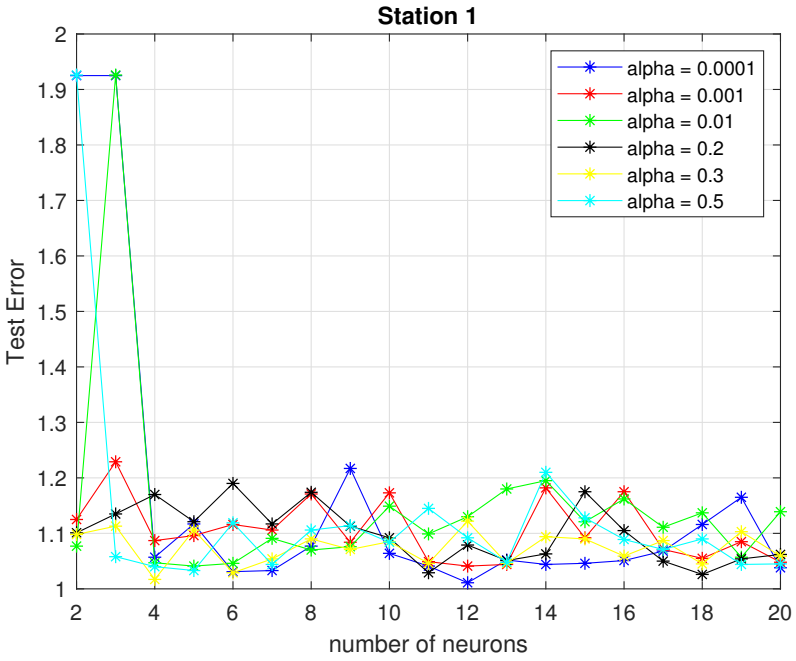


Figure 4.2: Test error as a function of the number of neurons (n) and regularisation value (α) on long term prediction

---

time interval 24 hours after the current time interval in the long term prediction task. The features used in this algorithm are the following ones:

1. Day_of_week (Integer from 0 to 6) - 1 feature
2. Is_holiday (Binary) - 1 feature
3. T_Cold (Binary) - 1 feature
4. T_Medium (Binary) - 1 feature
5. T_Hot (Binary) - 1 feature
6. Rain/Snow (Binary) - 1 feature
7. Hour (Integer from 0 to 23) - 1 feature
8. Minutes (Binary) - 1 feature
9. M_January (Binary) - 1 feature
10. M_February (Binary) - 1 feature
11. M_March (Binary) - 1 feature

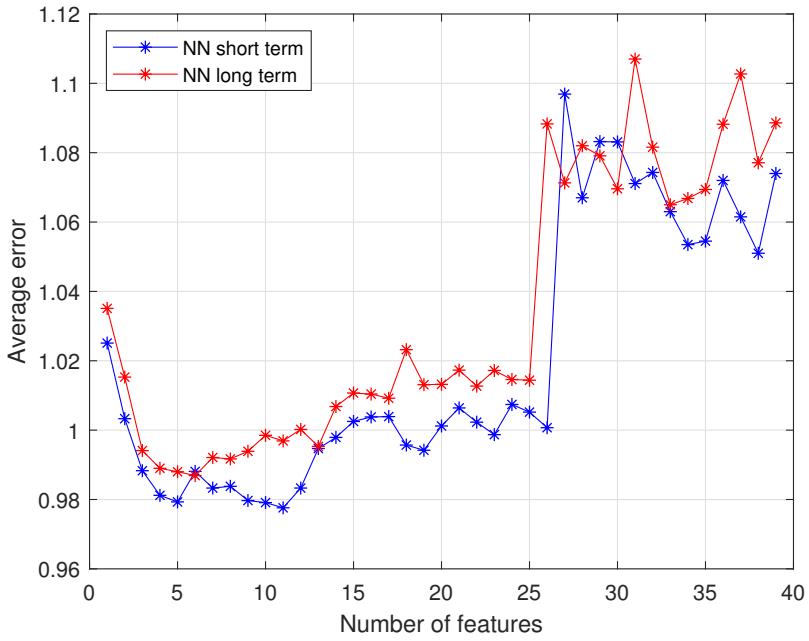


Figure 4.3: Average test error with different number of features (p) for short term and long term prediction tasks

---

12. M_April (Binary) - 1 feature
13. Is_today (Binary) - 1 feature

Concerning the Is_today feature (the last in the above list), this is a runtime feature whose value changes based on the current time instant in the following way: this feature is set to 1 on all historical time intervals of the same day as the current time interval, and it is set to 0, otherwise. The aim is to give more relevance to the historical data of the current day since their feature values (like the ones associated with temperature) are expected to be more relevant for the prediction.

Due to the fact that the efficiency of K-NN is penalised for large datasets, it was decided to cluster the dataset in different groups and, then, to use only one of the groups as input data to the algorithm. The 8 different groups are defined based on the values of 2 most important features (Day_of_week and Is_holiday), as ranked in Section 3.3.2. The resulting groups are:

1. All data from holidays
2. All data from Mondays except holidays
3. All data from Tuesdays except holidays
4. All data from Wednesdays except holidays

	n	α	p	*Test Error*
Station 2	14	0.01	7	0.799
Station 3	13	0.1	3	0.458
Station 4	9	0.1	3	0.264
Station 5	5	0.1	3	0.683
Station 6	2	0.5	3	0.607
Station 7	11	0.3	4	0.450
Station 8	11	0.0001	3	0.386
Station 9	13	0.5	3	0.645
Station 10	7	0.5	3	0.509
Station 11	13	0.0001	7	0.890
Station 12	12	0.1	3	0.291
Station 13	8	0.1	5	0.492
Station 14	2	0.3	5	0.577
Station 15	6	0.3	3	0.407
Station 16	6	0.3	3	0.442

Table 4.3: Study of neural network model parameters for short term prediction for the rest of stations

5. All data from Thursdays except holidays
6. All data from Fridays except holidays
7. All data from Saturdays except holidays
8. All data from Sundays except holidays

Then, the group of data that is given as input to the K-NN is the one whose Day_of_week and Is_holiday feature values are the same as the current time interval.

4.4.1 K-NN parameters

In order to get the best possible prediction accuracy, there are some parameters whose values/options must be selected. As explained in Section 2.5.4, the most important parameter is the number of neighbours (K). In this dissertation, the values of K between 1 and 5 were tested. The reason for limiting the tests to 5 is due to the small quantity of data. In the available historical data, the probability of finding many historical time instants similar to the current one is low and there is a higher probability of getting better accuracy if a small number of neighbours is considered (the analysis of results will confirm this assumption).

Another parameter that must be selected is the distance measure (d) to be used between data points. Two distance measures were tested. First, the *manhattan distance* which

	n	α	p	*Test Error*
Station 2	3	0.3	11	0.782
Station 3	6	0.5	3	0.465
Station 4	9	0.0001	3	0.256
Station 5	19	0.1	3	0.699
Station 6	16	0.0001	3	0.601
Station 7	3	0.5	4	0.440
Station 8	6	0.01	3	0.382
Station 9	5	0.5	3	0.638
Station 10	3	0.01	3	0.503
Station 11	10	0.1	3	0.881
Station 12	4	0.5	3	0.298
Station 13	14	0.001	4	0.488
Station 14	5	0.5	3	0.577
Station 15	7	0.01	3	0.407
Station 16	4	0.01	3	0.431

Table 4.4: Study of neural network model parameters for long term prediction in the rest of stations

defines the distance between two points as the sum of the modules of the differences of their corresponding dimensions (in this case, a dimension is a feature). The formula to calculate the *manhattan distance* between two points A and B in n-dimensions is the following:

$$D\_manhattan = \sum_{i=1}^{n} \mid A_i - B_i \mid \tag{4.2}$$

The second tested distance measure is the *euclidean distance* which defines the distance between two points as the square root of the sum of the squares of the differences of their corresponding dimensions. The distance between points A and B is calculated in this case by the following formula.

$$D\_euclidean = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2} \tag{4.3}$$

The last parameter that must be selected is the weight function (w) that determines how the output prediction value is computed from the K neighbours. Two weight functions were tested: *uniform* and *distance*. The *uniform* weight function assumes that all neighbours have the same influence in the prediction and computes the estimated value by a simple average of the neighbour values. The *distance* weight function gives higher weight values to the neighbours that are closer to the data point to predict and computes the estimated

value by a weighted average of the neighbour values. Note that this parameter w is only used when the number of neighbours K is higher than 1 since for the case of $K = 1$, the estimated value is given by the single neighbour value.

4.4.2 K-NN model selection for *Station 1* - short term prediction

Table 4.5 presents the test errors of the K-NN algorithm for all combinations of K, d and w. These results show that the minimum test error is 1.380 which is obtained with $K = 1$ neighbours and the distance measure $D\_euclidean$. Moreover, the results show that considering a larger number of neighbours K always decrease the method accuracy confirming in practice the previous assumption of not testing K values higher than 5. Nevertheless, the results also show that for each value of K higher than 1, the *distance* weight function has always better results than the *uniform* weight function while the impact of the distance measure is residual.

w	d	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
$distance$	$D\_manhattan$	1.380	1.439	1.500	1.531	1.538
$uniform$	$D\_manhattan$	1.380	1.654	1.783	1.847	1.905
$distance$	$D\_euclidean$	**1.379**	1.440	1.503	1.535	1.540
$uniform$	$D\_euclidean$	**1.379**	1.655	1.787	1.852	1.911

Table 4.5: K-NN Model selection (K neighbours, distance and weight function) on a short term prediction

4.4.3 K-NN model selection for *Station 1* - long term prediction

Like before, Table 4.6 presents the test errors of the K-NN algorithm for all combinations of K, d and w. These results show that the minimum test error is 1.425 which is obtained with $K = 1$ neighbours and with both distance measure options. Again, larger number of neighbour values K decrease the method accuracy. Finally, the results show again that for each value of K higher than 1, the *distance* weight function has always better results than the *uniform* weight function while the impact of the distance measure is residual.

4.4.4 Results for the Other Stations

The study previously described for *Station 1* was then repeated for all other 15 stations. Table 4.7 presents the K and d values that have provided the best test error and the obtained test error value for all other stations both in the short and long term prediction tasks. Note that since the best values are always with $K = 1$, the table does not include the wright function w. The analysis of these results are done in the next Section 4.5.

w	d	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
distance	*D_manhattan*	**1.425**	1.492	1.561	1.592	1.598
uniform	*D_manhattan*	**1.425**	1.673	1.806	1.902	1.927
distance	*D_euclidean*	**1.425**	1.492	1.561	1.588	1.594
uniform	*D_euclidean*	**1.425**	1.673	1.804	1.897	1.924

Table 4.6: K-NN Model selection (K neighbours, distance and weight function) on a long term prediction

	K	d (30 *min*)	*Error* (30 *min*)	d (24 *hours*)	*Error* (24 *hours*)
Station 2	1	*both*	1.185	*D_manhattan*	1.155
Station 3	1	*both*	0.734	*both*	0.702
Station 4	1	*both*	0.609	*both*	0.630
Station 5	1	*both*	1.035	*D_euclidean*	1.003
Station 6	1	*both*	0.952	*both*	0.884
Station 7	1	*both*	0.740	*both*	0.663
Station 8	1	*both*	0.699	*both*	0.696
Station 9	1	*both*	0.954	*D_euclidean*	0.941
Station 10	1	*D_euclidean*	0.795	*both*	0.806
Station 11	1	*both*	1.216	*both*	1.221
Station 12	1	*D_euclidean*	0.657	*D_euclidean*	0.598
Station 13	1	*D_euclidean*	0.746	*D_euclidean*	0.780
Station 14	1	*both*	0.860	*both*	0.866
Station 15	1	*both*	0.762	*both*	0.827
Station 16	1	*both*	0.761	*both*	0.735

Table 4.7: Study of K-NN model for short and long term predictions in the rest of stations

4.5 Analysis of Accuracy Results

Recall from Section 3.2.1 that *Station 1* and *Station 2* are the stations with more traffic in terms of number of pickups and drop-offs and *Station 11* is the one with the highest range of occupation. Therefore, one might expect that these three stations are the most challenging ones for the prediction methods. In fact, this is the case since all results of both ML algorithms and in both (short and long) prediction tasks show that these three stations are the ones with the highest test error values among all.

For the NN prediction algorithm, the results show that the test error values are always below 1.0 for all stations when the best parameter values are applied. The average test error among all stations is 0.56 for the short term prediction and 0.55 for the long term prediction. Since the usual station occupation values are between 5 and 15 bikes, these test error values can be considered of a good accuracy. Note that the accuracy of the NN

algorithm in the long term prediction task is similar to the accuracy in the short prediction task. This performance might be unexpected as the intuition is that the more in future we aim to predict, the hardest it would be.

For the K-NN prediction algorithm, the results show that test error values are always higher than the ones of the NN prediction algorithm for all stations even when the best parameter options are applied. The average test error among all stations is 0.88 for the short term prediction and 0.87 for the long term prediction. Again, given the usual station occupation values, these test error values can be considered also of a good accuracy but they are clearly worse than the accuracy of the NN models. Note that in the case of the K-NN, the test error values are more similar between different stations than they were in the NN case. Like in NN, the accuracy of the K-NN algorithm in the long term prediction task is similar to the accuracy in the short prediction task.

One of the reasons that might explain the inferior performance of K-NN (when compared with NN) is the fact that the used dataset is the historical data of only four months which are typically the ones with the lowest system usage (because they correspond to the Winter and initial Spring months). A larger dataset (i.e., with more months of historical data) has the potential of improving more significantly the K-NN prediction than the NN prediction since with larger datasets, the probability of past data points being closer to the time period in estimation (both in terms of distance and number of neighbours) increases significantly. So, K-NN with a K value higher than 1 will be potentially better for larger datasets.

Chapter 5

Conclusions and Future Work

The main objective of this dissertation was the development of a decision support service for Bewegen Bike-Sharing Systems (BSSs) based on Machine Learning (ML) methods to predict the occupation (in number of bikes) of each station of a Bewegan BSS. This objective had two aims defined as short term prediction task and long term prediction task.

The first step was to develop a backend data collection module to collect in a systematic way the Bewegen BSSs data needed to serve as input for the ML algorithms. Only after this first step being done, data started to be collected in 1 of January 2019.

The second step was to collect other relevant features like weather information and holidays information. These features contain information that was later concluded as being very important to obtain accurate predictions.

The third step was, based on all collected data, to determine which are the most important features that impact positively the prediction accuracy. The order of the features' importance was defined based on a well-known feature ranking process.

The fourth step was to implement and test two ML algorithms, one based on Neural Networks (NNs) and the other based on K-nearest neighbour (K-NN), for both prediction tasks (the short and long term predictions). The dataset used in this last step was the historical data collected from 1 of January 2019 until 30 of April 2019.

Concerning the analysis of the accuracy results, it was shown that the stations either with more traffic (in number of pickups and drop-offs) and/or more occupation range are the most challenging ones as the results of both algorithms and in both prediction tasks show that these cases have the highest test error values.

For the NN prediction algorithm, the average test error among all stations was 0.56 for the short term prediction and 0.55 for the long term prediction. On the other hand, for the K-NN prediction algorithm, the average test error among all stations was 0.88 for the short term prediction and 0.87 for the long term prediction. So, the NN based algorithm has better accuracy performance in both tasks but both algorithms can be considered of good accuracy since the usual station occupation values are between 5 and 15 bikes.

As future work, the most important task to be done is to run again the whole evaluation of both ML algorithms as soon as a complete year of historical data is available. This

task is important for three main reasons. First, a whole year of historical data will have more features (the number of months becomes 12 instead of the current 4). Second, the clustering of temperature values into the Cold, Medium and Hot categories will change as the limits between the categories are calculated based on the historical data (since the current historical data is of Winter and early Spring months, in a complete year dataset the limits will surely increase and current historical temperature values will be classified differently). Third, the usage of any BSS is typically much higher in the Summer season with higher occupation ranges in more stations (which was shown to be more challenging for the prediction algorithms). All these three factors might potentially affect the accuracy efficiency of both ML algorithms, the reason why their evaluation must be repeated.

Then, many other ideas come to mind when thinking about future work. One is to extend the evaluation of the developed ML algorithms to the available datasets of the other Bewegen BSSs (the data collection started in 1 of January 2019 is collecting data from all Bewegen BSSs). In this way, the accuracy of the implemented prediction tasks could also be studied to BBSs with different characteristics in terms of size, type of clients and geographical location. Second, state of the art suggests that other algorithms (such as gradient boosting decision tree or random forest) could also fit on this dissertation objectives and, so, their implementation and performance comparison with the implemented algorithms is a possible interesting future work.

Note that the decision support service developed in this dissertation is still not integrated in Bewegen internal system. In future, the short term prediction task has to be integrated in the website and mobile apps to be available to clients and the long term prediction task has to be integrated with the internal backoffice to be available to the system operators of each Bewegen BSS.

References

[1] Hangzhou information, https://https://use.metropolis.org/case-studies/hangzhou-china-urban-public-bicycle-sharing-program.

[2] Bixi montreal information, https://www.mironline.ca/a-review-of-2018-bixi-usage-in-montreal/.

[3] Capital bike-share information, https://en.wikipedia.org/wiki/capital_bikeshare/.

[4] Citi bike-share information, https://www.wired.com/story/lyft-bets-bikes-nyc/.

[5] A tale of twenty-two million citi bike rides: Analyzing the nyc bike share system, https://toddwschneider.com/posts/a-tale-of-twenty-two-million-citi-bikes-analyzing-the-nyc-bike-share-system/.

[6] Random forest, https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

[7] Artificial nueral networks, https://hackernoon.com/artificial-neural-network-a843ff870338.

[8] Support vector machine, https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be.

[9] K-means cluster, https://rpubs.com/cyobero/k-means.

[10] Elliot Fishman, Simon Washington, and Narelle Haworth. Bike share: A synthesis of the literature. *Transport Reviews*, 33(2):148–165, 2013.

[11] Danilo Saraiva. Servios de distribuio, recolha de dados e indicação de presença para a bicicleta bewegen, 2017.

[12] Xize Wang, Greg Lindsey, Jessica E. Schoner, and Andrew Harrison. Modeling bike share station activity: Effects of nearby businesses and jobs on trips to and from stations. *Journal of Urban Planning and Development*, 142(1):04015001, 2016.

[13] Ralph Buehler and Andrea Hamre. Business and bikeshare user perceptions of the economic benefits of capital bikeshare. 2015.

[14] Susan Shaheen, Hua Zhang, Elliot Martin, and Stacey Guzman. China's hangzhou public bicycle. *Transportation Research Record: Journal of the Transportation Research Board*, 2247:33–41, 12 2011.

[15] Catherine Morency, Martin Trépanier, Antonio Paez, Hubert Verrault, and Julien Faucher. Modelling bikesharing usage in montreal over 6 years. 2017.

[16] Vélib' métropole bike-share information, https://www.nouvelobs.com/societe/20180302.obs2970/le fiasco-des-nouveaux-velib-un-desaveu-pour-la-mairie-de-paris.html.

[17] Nicolas Gast, Guillaume Massonnet, Daniel Reijsbergen, and Mirco Tribastone. Probabilistic forecasts of bike-sharing systems for journey planning. 10 2015.

[18] Zidong Yang, Ji Hu, Yuanchao Shu, Peng Cheng, Jiming Chen, and Thomas Moscibroda. Mobility modeling and prediction in bike-sharing systems. pages 165–178, 06 2016.

[19] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. Traffic prediction in a bike-sharing system. *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 33:1–33:10, 2015.

[20] Haitao Xu, Jing Ying, Hao Wu, and Fei Lin. Public bicycle traffic flow prediction based on a hybrid model. *Applied Mathematics Information Sciences*, 7:667–674, 03 2013.

[21] R. Giot and R. Cherrier. Predicting bikeshare system usage up to one day ahead. *2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, pages 22–29, 2014.

[22] Hong Yang, Kun Xie, Kaan Ozbay, Yifang Ma, and Zhenyu Wang. Use of deep learning to predict daily usage of bike sharing systems. *Transportation Research Record*, 2672(36):92–102, 2018.

[23] Kyle Gebhart and Robert B. Noland. The impact of weather conditions on bikeshare trips in washington, dc. *Transportation*, 41(6):1205–1225, Nov 2014.

[24] Pengcheng Dai, Changxiong Song, Huiping Lin, Pei Jia, and Zhipeng Xu. Cluster-based destination prediction in bike sharing system. pages 1–8, 12 2018.

[25] Pierre Borgnat, Celine Robardet, Jean-Baptiste Rouquier, Patrice Abry, Patrick Flandrin, and Eric Fleury. Shared bicycles in a city: A signal processing and data analysis perspective. *Advances in Complex Systems*, 14, 06 2011.

[26] Steven R. Gehrke and Timothy Welch. A bikeshare station area typology to forecast the station-level ridership of system expansion. *Journal of Transport and Land Use*, 12:221–235, 04 2019.

[27] Lindsay Kathryn Maurer. Feasibility study for a bicycle sharing program in sacramento, california. 2012.

[28] Adele Cutler, David Cutler, and John Stevens. Random forests. *Machine Learning - ML*, 45:157–176, 01 2011.

[29] Gradient boosting vs random forest, https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80.

[30] Ming Zeng, Tong Yu, Xiao Wang, Vincent Su, Le T. Nguyen, and Ole J. Mengshoel. Improving demand prediction in bike sharing system by learning global features. *KDD 2016*, 2016.

[31] S.B. Imandoust and Mohammad Bolandraftar. Application of k-nearest neighbor (knn) approach for predicting economic events theoretical background. *Int J Eng Res Appl*, 3:605–610, 01 2013.

[32] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.

[33] Harris Drucker, Christopher Burges, Linda Kaufman, Alexander Smola, and V Vapnik. Support vector regression machines. *Adv Neural Inform Process Syst*, 28:779–784, 01 1997.

[34] Jerome Friedman. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38:367–378, 02 2002.

[35] Arnab Datta. Predicting bike-share usage patterns with machine learning, 2014.

[36] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[37] Harris Drucker. Improving regressors using boosting techniques. *Proceedings of the 14th International Conference on Machine Learning*, 08 1997.

[38] Susan Shaheen, Stacey Guzman, and Hua Zhang. Bikesharing in europe, the americas, and asia: Past, present, and future. *Institute of Transportation Studies, UC Davis, Institute of Transportation Studies, Working Paper Series*, 2143, 01 2010.

[39] Yilin Yan, Min Chen, Mei-Ling Shyu, and Shu-Ching Chen. Deep learning for imbalanced multimedia data classification. pages 483–488, 12 2015.

[40] Jaime Salvador-Meneses, Zoila Ruiz–Chavez, and José Rodríguez. Compressed knn: K-nearest neighbors with data compression. *Entropy*, 21:234, 02 2019.