TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Teoman Turan 194230IASM

Real-Time Prediction for Bike Terminal Availability

Master's Thesis

Supervisor: Sadok Ben Yahia
Professor
Co-Supervisor: Wissem Inoubli
Post-Doc Researcher

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Teoman Turan 194230IASM

Reaalajaline Prognoos Rattaterminali Kättesaadavuse Kohta

Magistritöö

Juhendaja: Sadok Ben Yahia
Professor
Kaasjuhendaja: Wissem Inoubli
Post-Doc Researcher

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Teoman Turan

03.01.2022

Abstract

This paper acquaints an improved idea with a prediction of the availability of bike-sharing terminals accusing terminals of high precision by using prior machine and deep learning techniques and by building a new approach based on them. The sample datasets that are utilized to prepare the deep learning model and to be trained to test accuracy contains genuine bike terminal information for the city of Paris, France and Oslo, Norway. All things considered, the framework can make it as convenient as conceivable subsequently it permits alterations later and can be applied to any city or spot without any problem. Processing the real-time location information of driver or user and historical data about bike terminal availabilities, the prediction engine predicts the availability of the bike-sharing terminals around and declares the terminal with the best availability at that time. To accomplish this result, a deep learning model which is based on Long Short-Term Memory (LSTM) was created and prepared with genuine datasets in the extent of this examination. Other than those, according to the product point of view, the framework in general uses appropriated frameworks to push forecast results to customer applications or administrations, which was determined by devouring datasets, in a solid way and with high accessibility. The developed framework offers some benefit to the clients who use a website or mobile application, where an interactive dashboard could take place and into which the developed prediction model is embedded, by assisting them to find the most accessible and the best available bike terminal on or around their planned path. Nowadays, bike sharing is increasingly used, therefore a prediction system with satisfying accuracy would be considered useful.

This thesis is written in English and is 64 pages long, including 7 chapters, 24 figures and 9 tables.

List of abbreviations and terms

AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
API	Application Programming Interface
ASTGCN	Attention-Based Spatial-Temporal Graphical Convolutional Network
AttConvLSTM	Attention-Based Convolutional Long Short-Term Memory
BSS	Bike-Sharing System
CGN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
CSV	Comma-separated Values
DL	Deep Learning
GB	Gigabyte
GCNN	Graphical Convolutional Neural Network
GPU	Graphical Processing Unit
HA	Historical Average
HTTP	Hypertext Transfer Protocol
ID	Identity
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
LASSO	Least Absolute Shrinkage
LSB	Least Square Boosting
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Square Error

PC	Personal Computer
PLSR	Partial Less Square Regression
RAM	Random Access Memory
RF	Random Forest
RNN	Recurrent Neural Network
SSD	Solid State Disk
SVM	Support Vector Machine
TAGCN	Temporal Attention Graphical Convolutional Network
TAZ	Traffic Analysis Zone
URL	Uniform Resource Locator
XGBoost	Extreme Gradient Boosting

Table of contents

List of figures

List of tables

1 Introduction

Bikes have kept their exceptional status within transportation systems for decades. Moreover, bikes have attracted more attention particularly in relatively large cities and metropolitan areas. Several factors are leading the growth in the vogue of bike usage around the world. Bikes have been propounded within the context of discussions about environmental issues such as arising air pollution, overconsumption of fuel oil, or noise pollution because they are considered environmentally friendly. In addition, they have the potential to mitigate traffic congestion risk, especially in metropolitan areas. Looking at the factors from a different standpoint, Bernhard states that bikes were in use and new bikes were sold as a rising trend during the COVID-19 breakout in 2020. As noted by Girardi, the general manager of Full Speed Ahead Europe, in the article, the demand for bicycles might have reared as swimming pools and gyms were closed for two weeks during the pandemic. It is also stated in the same article that people were motivated for bicycle usage by some governments through reward programs oriented to new bicycle buyers during the COVID-19 breakout. For example, in Italy, residents who bought an engineless vehicle were able to receive a €500 honorarium. For this program, €210m were allocated [1]. Like public vehicles and taxis, the growing demand for a bicycle as a vehicle has also given birth to bike hiring systems that allow people to utilize allocated bicycles around the city when needed for the payment. This is called bike-sharing, bicycle sharing, or bike-sharing system (BSS). A BSS in a city is built over a network of self-service bike stations, in other words, terminals that are composed of a series of docks. In this system, a bike is checked out by a customer for payment. A loyalty membership card or mobile application or credit/debit card can be used to complete the check-out process. After riding to the destination, the hired bicycle can be parked in a dock of another terminal/station in the network of the same BSS.

Figure 1. A terminal of Hubway (now called as Bluebikes) BSS in Boston [2] [3]

As a result of the growing importance of bikes, in parallel, bike-sharing has also gained attention in recent years around the world. Being easy-to-use and the availability of flexible transportation opportunities that can save the time of pedestrians, professional drivers, and tourists can be counted as two major reasons for the aforementioned demand rise. McDonnell states that there is a BSS in San Francisco that commenced in 2013, and as of 2015, around the city, there were 70 terminals [4]. It is also stated by Glusac that the size of the BSS of Indigo, a mass transportation company, in Philadelphia is estimated to be doubled within the upcoming 5 years, and it will occupy half of the electric fleet in the city. On the other hand, the Divvy BSS in Chicago, which supplied 3,500 more e-bikes in 2020, is supposed to operate 10,000 e-bikes by 2022. The main goal here is to scale the accessibility of the system up to 100 percent of the city [5]. Because bike-sharing systems have been drawing an augmenting interest all around the world and accordingly have been actively used by customers, being informed about the availability of a bike terminal station has been a crucial issue as well. Because it affects both customer decisions and bike allocation plans of bike-sharing companies.

1.1 Problem

The main problem which was on the target within the context of this work is how to find the most available and accordingly the most preferable bike-sharing station around the current location or particularly on the planned path of an end-user who can be a driver or a pedestrian. The key criteria to be considered here is the bike availability ratio of a bike terminal which is basically the percentage of available bikes to the number of free docks within the capacity of a station. The developed prediction engine shall make predictions for the availability of all active bike terminals falling into the range around the end-user, processing geolocation information and historical data as long as historical data, which is the main input of the engine, can be provided to the system.

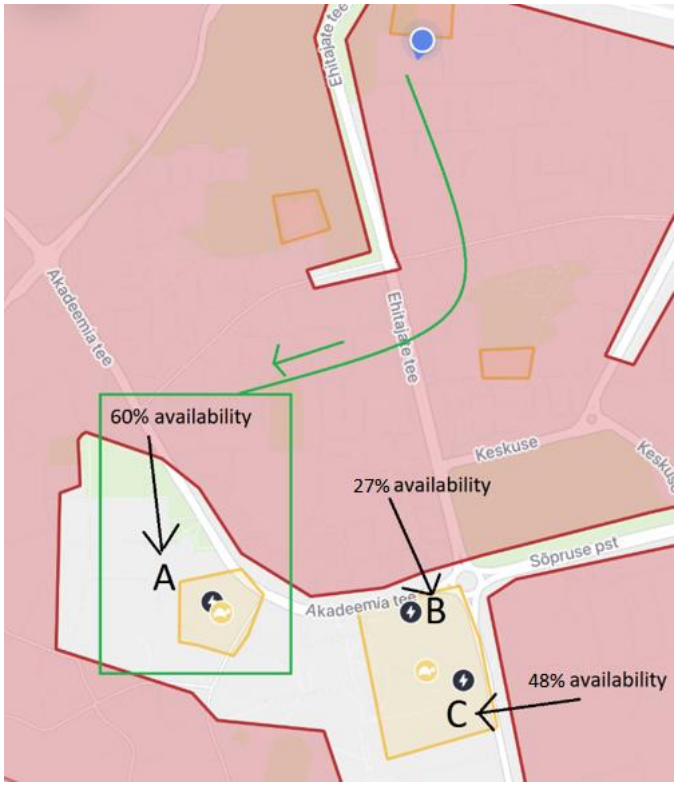

Figure 2. A simple mock-up about the problem and the proposed solution

1.2 Motivation

There are several motivations to study the aforementioned problem. To begin with, the developed system shall be dynamic, portable, and integrable to some extent, which means the prediction engine is proposed to make predictions for the bike stations in any region if historical data about bike terminal usage in that region can be fetched and to be used

within the back-end side of any integrable platform such as a desktop application, a webpage, or a mobile application. Such an easy-to-reach and easy-to-use platform can boost the awareness of end-users on the utilization of bike-sharing. The more a bike-sharing system in a city is used, the less traffic congestion and even air pollution are expected to exist around the affected region.

1.3 Contributions

Within the context of the thesis, there are two major contributions.

- A deep learning model that outperforms state of the art and some common machine learning and deep learning models in terms of accuracy was developed. It forms the basis of the prediction engine; hence it will help the system outcome relatively good predictions regarding bike station availabilities.

- The prediction engine proposed to be developed has a dynamic structure so that it can be used in any region for which historical data about bike terminal usage can be fetched and processed. Pieces of training were done over a limited number of datasets from different cities, but ultimately, they fed the development process of the model.

1.4 Paper Structure

The paper was written in English. The paper begins with two cover pages, one is in English and the other one is in Estonian. Then an abstract in English, page index and lists of abbreviations, tables, and figures take place. The main content of the paper begins with the introduction chapter, the first chapter, where the problem, motivation, and paper structure are mentioned. The second chapter is the background part where some preliminary information about machine learning and deep learning are given and a literature review to see the state of the art regarding the main problem is provided. The third chapter is the data part where datasets used to train the developed model, how they were scrapped and built, and how they were processed are explained. The fourth chapter is about the developed prediction model. In this chapter, details about the developed model, differences with the baseline model, and an evaluation of the prediction accuracy benchmark take place. The fifth chapter covers software development environments and

tools used within the scope of this study and the architecture of the proposed application. The sixth and final chapter is dedicated to conclusions regarding the study and some information about work that shall be done in the future. The paper ends with a list of references and appendixes.

2 Background

As mentioned before, to solve the said prediction problem, a deep learning model which is an LSTM-based solution was developed. Before explaining the development process in detail within the scope of the study, the background must be clarified. Therefore, some brief information about what machine learning and deep learning are and state of art covering the status of the results from the recent studies on the problem must be provided.

2.1 Machine Learning and Deep Learning

Artificial intelligence (AI), the ability of a computing machine to simulate human attitudes to solve complex problems, has several subfields. Machine learning (ML) is one of the ways to use AI in a computer system. This subfield concerns how to teach a computer to solve complex problems without being particularly programmed for them. Within this context, essentially, computers are trained to accomplish complex tasks in their effort by experience with historical data. Today, there are manifold applications of ML. For example, a computer can be trained to recognize the pictures of several dissimilar people. It would be quite time-wasting to develop an application that teaches the computer the face pictures one by one. It could be done at a high pace by human beings, but when it comes to expecting a satisfying performance from a computer, it can be considered tough to let the machine memorize each picture. Hereat, an ML approach can let the computer learn to program itself based on processed experience data, to complete the objective faster on behalf of the human effort it imitates. On the other hand, ML can be used to make predictions in brief. There are several ML models and algorithms to be used today to make predictions, each one being effective on different data types, taking different times, and resulting in different performances. Concerning data and what to be estimated, a programmer decides on an ML model to be used, then provides data as a set of input to it and runs it to train the computer itself. As a result, the system outcomes a prediction that has been generated by extracted patterns based on the training data. Accuracy is calculated by some metrics to observe how far actual data and predicted data are from each other. The less error means the better accuracy. The amount of data to be supplied to the ML model is also significant; the more data the computer is trained with, the better accuracy the model results in. Additionally, existing ML models and algorithms

can be tuned up by modifying structure or parameters by programmers too, to yield better accuracy benchmarks. [6]

As another real-life example for ML models, decision trees can be taken into account. To solve the problem of predicting house prices, the data set below can be supplied to the decided ML model to be processed to train the computer that is expected to estimate the price of an upcoming house.

ID	**Room number**	**Price**
1	4	€250,000
2	2	€150,000
3	3	€200,000

Table 1. Example simple data set for house price prediction

Here, the question "Is the room number less than, greater than, or equal to 3" can be put into the centre of the training to be used for making a decision. If the upcoming house has 3 rooms, its price is predicted to be €200,000. If it has 5 rooms, since 5 is greater than 3, its price is predicted to be €250,000 by the ML model.

Deep learning (DL) is an important subset of machine learning. It is fed by artificial neural network layers that are modelled and configured to operate like human brain cells and to process huge amounts of data. One of the important features of DL is that multiple data sources can feed inputs of models that analyse in real-time. As a DL model should be able to fetch data from multiple sources simultaneously, there are graphical processing units (GPUs) particularly optimized for DL training models. Compared to ML, more complex tasks can be accomplished by DL and new features can be produced by DL models by themselves. Today, DL is used for several fields such as social media where user profiles are extracted and advertisements are shaped based on user data like a large number of images, finance where stock values are estimated and trading strategies are developed, healthcare where patient sicknesses are guessed, cybersecurity where new threatening and suspicious activities can be recognized, or digital assistants like Siri, Cortana, Alexa where natural language processing is utilized [7].

2.2 State of the Art

There have been several studies over the main problem of the paper, bike terminal availability, over several years. It means that research about the problem has not been over yet, and new solutions have been provided to have ascended performances gradually. Essentially these studies differ from each other by several aspects such as used methods, accuracy performance, or dataset structure. Within the context of a literature review about state of the art, the most significant factor to take into account is accuracy performance that is measured by some common metrics such as mean absolute error (MAE), since its enhancement is one of the main objectives to fulfil in the ongoing studies.

Xu, Ji, and Liu conducted a study about bike demand prediction for station-free bike-sharing systems [8]. The covered region in this study was the downtown area of Nanjing City, the capital city of Jiangsu province on the east coast of China.

One of the main contributions of this study was to work on a bike-sharing network that is not station-based. Essentially data processed within the scope of the study are time-series data which are constituted by measurements over time. On this wise, another main contribution of this study is the tuned-up deep learning approach that is based on Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) and outperformed conventional time-series prediction models and machine learning algorithms. According to the authors of the paper, conventional artificial neural network (ANN) does not account for temporal dependencies in the model structure, that is why it cannot completely handle time-series data characteristics. A recurrent neural network is preferred as it is a feed-forward neural network structure. Feed-forward neural networks perform at satisfying benchmarks on time-series data. On the other hand, very long-time lags cannot be handled by conventional RNNs, therefore LSTM RNN was claimed to fit the time-series data by the authors of the paper. It can also be counted among the main contributions of the study that the developed dynamic forecasting model has the potential to improve the operational effectiveness of the aforementioned station-free bike-sharing system in the city and to rebalance the system by forecasting demand gaps.

Time-series data covering station-free bike sharing usage in the city were collected for one month through a crawler. The lowest missing rates were detected to exist in the consecutive 14 days between June 19th, 2017 and July 2nd, 2017. Eventually, the built

dataset consisted of 0.178 billion geolocation data. An example fetched data JSON that corresponds to a row in the dataset is as below:

```
{
      "2017/06/23 21:37:34",
      "object": {
            "bikeIds": "0250045914#",
            "distX": 118.73578455079466,
            "distY": 32.04531884191313
      }
}
```

Figure 3. Example JSON object referring to a record for Nanjing City

Field	**Description**	**Example**
Timestamp	Timestamp of the record creation	2017/06/23 21:37:34
bikeIds	Bike ID The first field in the sub-object "object"	0250045914#
distX	Longitude of a bicycle The second field in the sub-object "object"	118.73578455079466
distY	Latitude of a bicycle The third field in the sub-object "object"	32.04531884191313

Table 2: Dataset Model for the Station-Free Bike Sharing Network in Nanjing City

The authors also claim that exogenous data which are weather data, air quality data, and land-use patterns were also collected to boost the accuracy benchmark. Hourly historical weather data were fetched from Nanjing Meteorological Bureau and the hourly air quality data were fetched from the National Environmental Monitoring Centre of China. Finally, point of interest data was fetched from the Baidu map API to generate land-use patterns.

Before going on processing the dataset built up by records like above, the authors of the paper first divided the city into traffic analysis zones (TAZs), each one being 1 km^2. Eventually, there were 118 TAZs. For each TAZ, trip production and attraction models for bicycles were generated for four different time intervals: 10 minutes, 15 minutes, 20 minutes, and 30 minutes. A GIS tool was used to match each TAZ with trip data for trip production and attraction models as well as to calculate the centroid distance between each adjacent TAZs.

Figure 4. Nanjing downtown area divided by TAZs

Data covering the first 10 days were chosen to be the training dataset and those covering the last 4 days were chosen to be the validation dataset, in this study. The built dataset with trip demand data along with external data which are weather, air quality, and the land of use were then normalized, encoded, and undergone a fusion process to get ready to be processed and used for prediction by the developed deep learning architecture. Here, the developed deep learning architecture is basically based on LSTM RNN as mentioned. But to ascend accuracy performance, a basic LSTM RNN implementation was tuned up by altering some critical parameters. The model optimization, called sensitivity analysis, was carried out to tune up four parameters of LSTM RNN: training epoch number, batch size, node number, and dropout rate. The Adam optimization algorithm was used, and the aforementioned parameters were set to 80, 600, 60, and 0.07 respectively.

Within the context of evaluating predictive performance of the tuned-up LSTM-based model in this study, there are several findings. First of all, the more various type of data

the processed dataset had, the lower MSE scores were. For example, for the 30-minutes interval in a TAZ, if there was only trip data, the MSE for the production model was measured to be 28.374 and that for the attraction model were measured to be 25.875. But if full data (trip data + weather data + air quality data + land use data) was used, the MSE for the production model was measured to be 12.542 and that for the attraction model were measured to be 16.979. On the other hand, it was concluded that the number of selected nearest TAZs around a TAZ affected the performance of the TAZ. The more TAZs were taken into consideration, the better the accuracy benchmark could be accomplished. But the thing is the performance enhancement stopped once the number of TAZs were reached 5. In the light of this information, it can be said that it was impossible to boost the accuracy when the number of selected nearest TAZs was greater than 5. Finally, for the sake of comparison, the same dataset was trained by some other approaches, comprising one-step forecast, historical average (HA), artificial neural network (ANN), support vector machine (SVM), extreme gradient boosting (XGBoost), and autoregressive integrated moving average (ARIMA), as well, for the same time intervals and once for production model and once for attraction model. It was observed that the developed LSTM-based model outperformed all other approaches training the same dataset for the same time intervals and data models.

Another study conducted by Ashqar et al. used the approaches least-square boosting (LSB), partial less-square regression (PLSR), and random forest (RF) over a dataset covering the San Francisco Bay Area Bike Sharing System at network and station levels [9]. The main problem of the study was to estimate the availability of each station. In this study, RF and LSB were considered univariate regression algorithms to forecast the number of available bicycles at a station whereas PLSR was considered a multivariate regression algorithm to regenerate the spatiotemporal interactions between stations in the system to cover the entire network in the target area. It was eventually found that prediction errors of univariate models were lower than that of the multivariate model. But it should also be considered that the univariate models, RF and LSB, can be used for station-level whereas the multivariate model, PLSR, can be used for network-level where plenty of stations are interconnected and spatially correlated. It was also concluded that environmental variables such as demographics, neighbours, prediction horizon times notably impacted the accuracy performance at the station-level prediction.

Database trained within the context of this study was formed by combining multiple datasets, one being about trips as anonymized, the other one being about stations, the other one being about historical weather records, and the final but the most important one being about station status history. Datasets consist of data having been fetched from the time frame between August 2013 and August 2015 in the San Francisco Bay Area. The said ultimate database can be found at https://www.kaggle.com/benhamner/sf-bay-area-bike-share as well. Briefly, the station status history dataset has the following columns.

Field	**Description**	**Example**
station_id	Unique ID of the station	2
bikes_available	The number of available bikes at the station at the recorded time	2
docks_available	The number of available docks at the station at the recorded time	25
time	Record timestamp	2015/07/29 12:05:03

Table 3. Dataset structure for station status history records in the San Francisco Bay Area

The “time” column of the dataset was then extracted into subfields as year, month, day-of-month, day-of-week, time-of-day, and minute. Each station status history data was combined with daily weather information as well. The weather dataset has the following information: day (e.g. 9/1/2015); ZIP code of the address of a station; wind direction degree; maximum, mean, and minimum temperatures, dew points, humidity, sea level pressures, visibilities, wind speeds; maximum gust speed; precipitation level; cloud cover; events (whether it was foggy, rainy, or sunny).

When it comes to accuracy performance evaluation, it was clarified that RF slightly outperformed LSB, because the mean absolute error (MAE) for RF was 0.37 bikes/station whereas that for LSB was 0.58 bikes/station. On the other hand, these univariate models outperformed the multivariate model, PLSR, since the mean absolute error score for PLSR was 0.6 bikes/station. (The lower MAE training scores, the better accuracy it has.)

As another research over the same problem, Liu et al. worked on LSTM covering a bike-sharing docker in Suzhou, China [10]. The main problem, how to predict the bike availability at the docker at a given time step, is basically a time series analysis problem. Data coming from former time steps can help the system predict the bike availability at the upcoming time step. The authors of the paper claimed that they had chosen LSTM considering it the best model to solve a time series problem. In this study, apart from a basic or standard LSTM implementation (baseline), they presented an improved LSTM-based model as well.

The dataset used to train models were relatively simple, compared to other datasets observed within literature review over state of the art. Because it covers the history of only one bike station for 1 month. The dataset that can be reached at http://resuly.me/data/bike_rnn.csv has the following columns:

- Number of available bikes per minute
- Day of the week (i.e. 1-7)
- Hour of the day (i.e. 1-24)

The dataset has 45959 rows, each one corresponding to a minute. The dataset was first converted into sequences, each sequence length is 20. Afterwards, there were 45930 sequences. 95% of them were used for training whereas the remaining 5% of them were used for testing.

The first approach, standard LSTM implementation, is relatively simple. It has two layers, each one having 19 time steps. The simplest architecture, benchmark design, each time step has only one input, the number of available bikes at the current step, and only one output, the number of available bikes at the 20^{th} step. The authors of the paper proposed two new architectures for the sake of better accuracy performance for this approach: multi-features design where two input features, the day of the week and the hour of the day, were added and multi-output design where outputs were extracted from not only the last step but also the penultimate two steps. Above all, it was also claimed that the more features a model has, the higher accuracy the model results in.

The second approach is an LSTM-based implementation again, but it was developed by the authors and can be considered relatively complex. To make better estimations, it was claimed to be a decent way to know what would happen multiple time steps later. Within the context of this approach, data generation and prediction were considered two main phases. In the data generation phase, time steps from 0 to the last one (except this) in a row were used to predict the last time step value, then the first step was dropped.

In the end, for the baseline (one-time step prediction), it was found that the MAE score for the standard (benchmark) architecture had the highest one whereas that for the multi-output architecture had the lowest one. On the other hand, the developed multiple time steps prediction approach outperformed the baseline in terms of the MAE scores.

Zi et al. also proposed another approach that is based on a novel deep graph convolutional network (CGN) model with temporal attention (TAGCN) in their studies where four seasons data of Divvy Bike Sharing System in Chicago was used [11]. The main contribution of this study was to rebalance the number of available bikes at different stations regularly. Station-level demand estimation played an important role in the said rebalancing work too. On this wise, the bike check-out/bike check-in ratio of each station was estimated by GCN. The impact of different time granularity was reflected by the model as well.

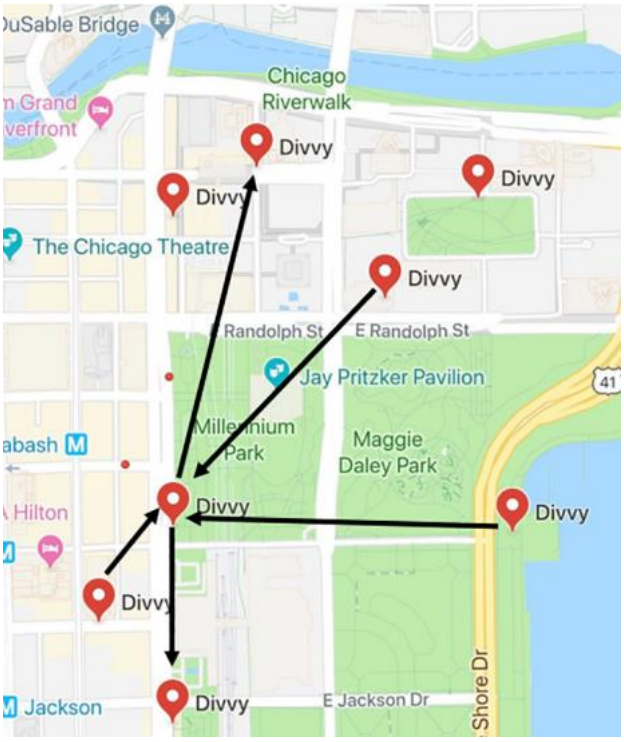

Figure 5. Mock-up about bike check-out/in of a target station where an arrow indicates the move between two stations, for TAGCN

Four different datasets were built up, each one corresponding to a different season. The structure of each bike-sharing station was basically a graph. Flow data of bike stations of the aforementioned bike-sharing system were counted every 30 minutes. In addition, real-time data from the bike-sharing system were combined with the historical weather data. The head of 60% out of data was used for training and the last 20% out of data were used for testing.

The developed TAGCN model was compared to some baseline models as least absolute shrinkage and selection operator (LASSO), LSTM, ConvLSTM, AttConvLSTM, historical average (HA), SVM, GCNN, ASTGCN, and STG2Vec, in terms of accuracy performance. Eventually, it came to the conclusion that the proposed TAGCN model outperformed all other techniques for all four seasons by MAE.

2.3 Summary

State of the art about the main problem, estimating bike terminal availability, shows that a variety of techniques have been used so far, but some of them are relatively popular.

For example, LSTM has been detected to success if the dataset covers time-series data. Moreover, the type of data makes sense on the selection of the baseline approach that is to be tuned up or developed. By altering the hyperparameters or the structure of a model, it can be tuned up so that it can yield better accuracy. A successful tune-up has the potential to lead the way to outperform other techniques being run over the same dataset. On the other hand, reaching a satisfying accuracy should not be considered enough. A comparative study between different approaches must be carried out, and the developed model must result in a higher accuracy benchmark than the state of the art.

3 Data

One of the important parts of the implementation within the context of the study in this paper is input data. As mentioned before, the prediction engine whose core is operated by the prediction model that was developed in this study must receive historical data to process them and to make predictions. As stated in the “2 Background” part of the paper, an ML/DL prediction model evaluates patterns formed by processed historical data and makes a guess for an upcoming entry. At this point, it can be inferred that estimations are the outcome of an algorithm that is fed by historical data submitted to the system as input. In this paper it was also stated that the proposed engine within the proposed architecture is considered to be dynamic. Therefore, as long as historical data about the usage of a bike terminal or bike terminals in a specific zone are provided to the system, regardless of the data source, the system shall predict bike terminal occupancies in the area.

The data input of the proposed architecture is basically receiving a dataset that is composed of historical record rows. The received dataset is then sent to the prediction engine to be processed and normalized to get ready to be used for prediction first. Then the converted dataset containing historical data is used by the model lying in the core of the prediction engine to create historical patterns and to predict bike terminal occupancy ratio, in another word, availability. Hereby, from the perspective of prediction technique development within this study, the model must be trained with at least two different datasets for the sake of the verification of accuracy benchmark having reached a satisfying level.

The input of the architecture was designed to receive datasets that are stored in comma-separated value (CSV) files. In other words, the prediction engine reads historical data from a CSV file. Johnson states that a CSV file is a basic text file where information is concatenated by a delimiter and stored in this way. A CSV file is composed of rows and each row comprises sequential separated information in the same pattern. To open a CSV file to see the content, any text editor or spreadsheet processing application such as Notepad or Microsoft Excel can be used. Even though the name of the file type implies it, the delimited used in a CSV file can be a character different than comma (,), such as space or semicolon, too. However, comma is the most common delimiter used to separate stored information in a CSV file [12].

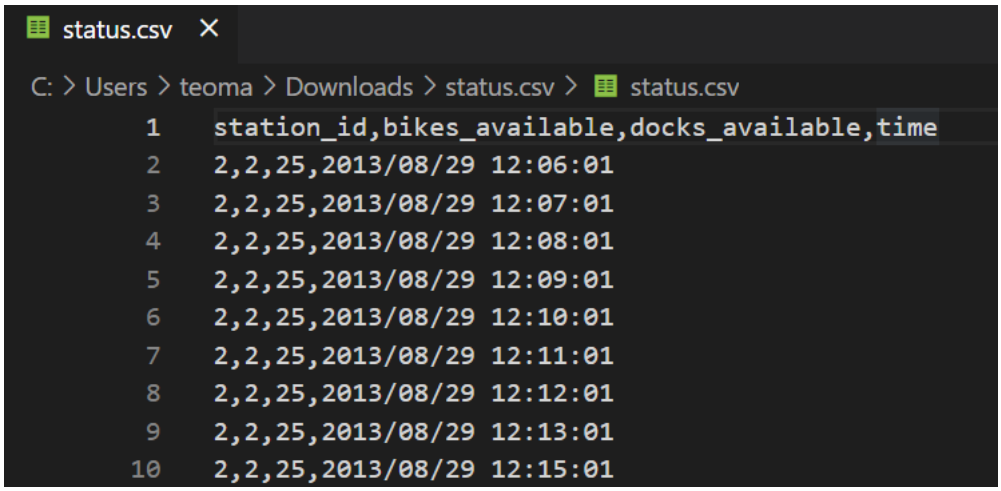

Figure 6. An example CSV file opened with Microsoft Visual Studio Code

In the sample CSV file in Fig. 4, where historical data about bike terminal availability are held, the first row holds column titles whereas the rest of the rows hold bike terminal availability data themselves. For example, the second row means that there were 2 bikes available and 25 docks available at station 2 at 12:06:01 on August 29th, 2013.

It is also possible to view a CSV file in a decent table by opening it with a spreadsheet editor such as Microsoft Excel.

3.1 Data Sources

Ultimately, while the prediction technique was being developed in this study, to evaluate its accuracy benchmark, a couple of datasets that are stored in separate CSV files were built and used to train the model and to compare the performances of models over them to each other.

- A dataset for Vélib' BSS in Paris, France
- A dataset for a bike terminal in Suzhou, China
- A dataset for City Bike BSS in Oslo, Norway

3.1.1 Vélib' BSS in Paris, France

The first dataset holds historical data referring to occupancies of bike terminals of the Vélib' BSS in Paris, the capital city of France.

Vélib' BSS commenced their operations in Paris in 2007. It is considered the avant-garde of bike-sharing systems around the world. Today, Vélib' is one of the primary public transportation facilities in the city. In the Greater Paris area, there are 1400 Vélib' bike terminals (dockers) where 20,000 bikes, 40% out of them being electrical, are hired and operated. On the other hand, in 2020, there were 400,000 subscribers of the system. The daily record of the system was broken on September 11th, 2020, with 215,000 trips. In addition, the monthly record of the system was broken in September 2020 with 5.5 million trips [13]. On the other hand, 8 years ago Jacobsen stated that the traffic density in the Greater Paris area was expected to decrease by 40% by the year 2020 as a result of the growing integrating of Vélib' while a 20% reduction had already been achieved since the commencement of the service. The expenses of the BSS are met by JCDecaux through contract-based advertisements [14].

Figure 7. A Vélib' bike terminal in Paris [15]

3.1.2 Suzhou, China

The second dataset holds historical data referring to not multiple bike terminals of a bike-sharing system, but a certain bike terminal, in Suzhou city of China. This dataset has actually been already addressed in the section “2.2 State of the Art” in this paper because it was used in the study [10]. The same dataset was used for the experimental training process as well.

3.1.3 City Bike BSS in Oslo, Norway

The third dataset holds historical data referring to occupancies of bike terminals of the City Bike BSS in Oslo, the capital of Norway.

Similar to the Vélib' BSS in Paris, Oslo City Bike (“Oslo Bysykkel” in Norwegian) is supposed to be a significant element of the public transport in Oslo. The BSS was used approximately by 100,000 people in Oslo who made more than 2.7 million bike travels in 2018. The BSS is a product of a collaboration between Oslo city management and Clear Channel Norway AS. The expenses of the system are met by agreement-based advertisements and sponsorships. The development of the system is handled by Urban Infrastructure Partner Oslo Bysykkel AS. The thing is the system is available for use within a limited timeframe. Within the declared scheme, bicycles of the BSS can be unlocked within the time frame from 5 AM in the morning to 1 AM next night. Apart from bike-sharing functionality, they also offer bike repair and cleaning services. End-users can use the system via a mobile app [16].

Figure 8. A bike terminal of Oslo City Bike BSS [17]

3.2 Dataset Building

Dataset building is one of the most significant processes of implementation in this study. Among the addressed data sources in the previous section of the paper, the aforementioned process was carried out to build a dataset for the Vélib’ BSS in Paris and to build a dataset for City Bike BSS in Oslo. (The dataset for a certain bike terminal in Suzhou was already provided as built.)

The main purpose of building a dataset is essentially to hold real-time statistics of bike terminal usage (available bike number, available dock number, the status of the terminal etc) per a unit time such as a certain number of seconds. Apart from these statistics, a dataset can also contain additional information about bike terminals such as their geocoordinates (longitude, latitude) or addresses.

The data sources mentioned in this study already store historical data in their databases. There are also provided open data APIs to retrieve data from those databases. The thing that was done within the implementation is to fetch data and reflect them into a CSV file in a neat way. In general, the dataset building process consists of the following steps.

- Step 1: A function that sends an HTTP GET request to the relevant API to be called to fetch the last *n* records from their database, where *n* can be sent as a parameter in the request URL, is created. Here, the "*requests*" library of the Python programming language can be used to send an HTTP GET request.

- Step 2: A job that executes the function defined in the previous step every *n* minute is created, then is started. Here, either a *cron* job can be defined for a Linux distribution or macOS or as a cross-platform solution that can work on Windows or any other operating system as well, the "*BlockingScheduler*" library of Python can be used. The interval definition of the job is set to *n*.

- Step 3: HTTP GET requests to return a response as an array of JSON per each call. Here, to achieve this, the "*JSON*" library of Python programming language can be used.

- Step 4: Each element of the JSON array response having been retrieved is then parsed. Basically, if a blank CSV file for a dataset has just been created, the first row to be written down consists of column titles statically. Otherwise, if data is attached to a non-empty CSV file, writing column titles down is skipped. Each element of the JSON array basically corresponds to each row to be written down in the CSV file. Iterating through the elements of the array in a loop, each JSON is parsed, then all meaningful elements of the JSON are written down in the CSV file one by one. Eventually, the CSV file is filled with a "dataset" by the system. For this purpose, the "*CSV*" library of Python programming language can be used.

- Step 5: All steps above are repeated by the defined and commenced job scheduler. Accordingly, the size of and the number of the records in the dataset CSV file rise. In order to halt the system assuming enough data have been accumulated, simply, the scheduler can be stopped.

Going over the generic steps above, to build a dataset for Véib' BSS in Paris, France, an API provided by a website named after Paris Data was used [18]. The website basically follows the Open Data approach of France that refers to the accessibility of legal information from an online source. On this website, it is possible to find several datasets covering statistical and historical data published by the city government of Paris and their partners. Datasets are licenced under the Open Database License (ODbL) license as well [19]. The dataset shared on the website to get real-time historical data about bike terminals of the aforementioned BSS is "Vélib - Bikes and terminals - Real-time availability", originally called "Vélib - Vélos et bornes - Disponibilité temps réel" in French. It can be reached at the URL below.

https://opendata.paris.fr/explore/dataset/velib-disponibilite-en-temps-reel/information/?disjunctive.name&disjunctive.is_installed&disjunctive.is_renting&disjunctive.is_returning&disjunctive.nom_arrondissement_communes

Paris Data provides a table of the latest real-time data of the dataset that are presented neatly on the webpage, as can be seen from Fig 8. There are also some functionalities such as marking bike terminals on the map, data analysis, visual demonstration of statistical data, exporting facility to export the dataset into various file types such as a CSV file, a Microsoft Excel file, or a JSON file.

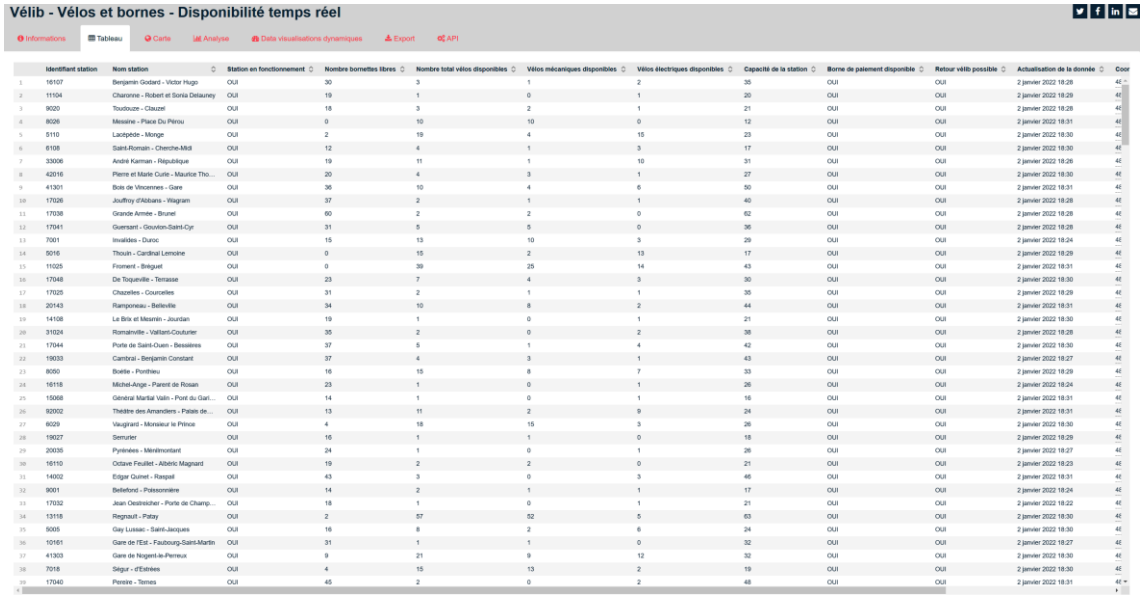

Figure 9. The table of the latest real-time data for the Vélib' BSS in Paris on the webpage of Paris Data

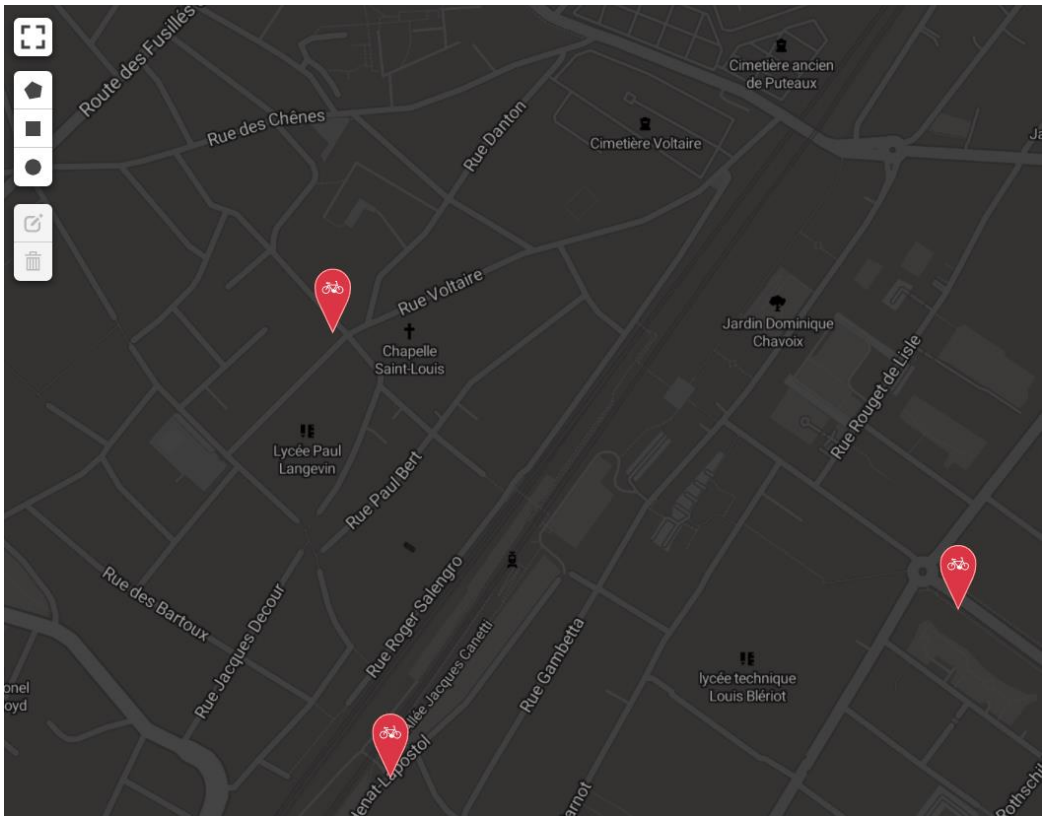

Figure 10. Locations of some of the Vélib' bike terminals on the map, provided by Paris Data

For this research, the most important thing is the provided API which can be found from the "API" tab on the same page. The API is actually named after "Search API" and it was established by an outsourcing company, Opendatasoft. There is also detailed documentation regarding the API [20].

After comprehensively embracing the API, to start retrieving data and building the dataset, the provided API was called by HTTP GET requests sent through the following URL:

```
https://opendata.paris.fr/api/records/1.0/search/?dataset=velib-disponibilite-en-temps-reel&q=&rows=2000
```

There parameter sets of the URL could be richer, e.g. a column in the dataset could be filtered or the beginning timestamp could be stated, but within the context of the research on this paper, two important parameters were used, as can be seen from the URL:

- `velib-disponibilite-en-temps`, that is the ID of the dataset

- `2000`, the number of rows or the number of latest records to be fetched at once

The response of this call is, as mentioned, an array of JSON objects. The array is basically stored in the parent element "`records`", and each element of the array is a JSON object like the one below.


```
"datasetid":"velib-disponibilite-en-temps-reel",
"recordid":"ce4cec0eb74c89934fa4a5df793b1f4d06b1eb1e",
"fields":{
      "name":"Benjamin Godard - Victor Hugo",
      "stationcode":"16107",
      "ebike":2,
      "mechanical":1,
      "coordonnees_geo":[
            48.865983,
            2.275725
      ],
      "duedate":"2022-01-02T16:28:36+00:00",
      "numbikesavailable":3,
      "numdocksavailable":30,
      "capacity":35,
      "is_renting":"OUI",
      "is_installed":"OUI",
      "nom_arrondissement_communes":"Paris",
      "is_returning":"OUI"
},
"geometry":{
      "type":"Point",
      "coordinates":[
            2.275725,
            48.865983
      ]
},
"record_timestamp":"2022-01-02T17:25:00.355000+00:00"
```


Figure 11. An example JSON object standing for a historical record for the Vélib' BSS in Paris

It can be understood that a JSON object in the array is a JSON-serialization reflection of a row in the dataset plus some additional information. Each field of a JSON object corresponds to a column in the table. The response of the call was then parsed and deserialized, and each field of an object in the array was then written down into the dedicated CSV file.

The dataset builder for the Vélib' BSS in Paris was executed by a job scheduler with an interval of 30 minutes for a limited amount of time. Eventually, more than 100,000 rows were accumulated in the CSV file. This is called a raw dataset. It was then preprocessed and normalized to get ready to be used by the prediction model.

A similar approach was carried out for the City Bike BSS in Oslo, Norway. The BSS provides an API to retrieve real-time data from their database. It is stated that every 10 seconds, a new record is added, and the dataset is accordingly updated [21]. An open data licensing approach is present here as well. With respect to this standard, the system broadcasts data under the Norwegian License for Open Government Data (NLOD) 2.0 [22]. The documentation of the API is also available in their GitHub repository [23].

Similar to the API for the Vélib' BSS in Paris, the API of the BSS must be called. There are three endpoints that can fetch data from the database of City Bike: system, stations, availability. The end-point URLs of the API of the City Bike BSS do not take any parameter. These datasets can be fetched via the following HTTP GET request URLs respectively.

```
https://gbfs.urbansharing.com/oslobysykkel.no/system_information.json
```

```
https://gbfs.urbansharing.com/oslobysykkel.no/station_information.json
```

```
https://gbfs.urbansharing.com/oslobysykkel.no/station_status.json
```

The responses are JSON object arrays consisting of JSON elements that can be parsed, in a similar way to the Vélib' BSS in Paris.

A JSON element within the system information response is like the one below. But this information did not make any sense to the study in this paper. Therefore, it was not used to attach to the main dataset. However, it can be used to get information about the system and the status of the system.


```
{
        "last_updated": 1553592653,
        "ttl": 10,
        "data": {
           "system_id": "oslobysykkel",
           "language": "nb",
           "name": "Oslo Bysykkel",
           "operator": "UIP Oslo Bysykkel AS",
           "timezone": "Europe/Oslo",
           "phone_number": "+4791589700",
           "email": "post@oslobysykkel.no"
        }
}
```


Figure 12. An example JSON object in the response of system information endpoint of the API of the City Bike BSS in Oslo

The response from the call to get information about stations contains an array of JSON objects, each one standing for a bike terminal of the BSS. A JSON element in the "`stations`" array is like the one below.

```
{
        "station_id":"627",
        "name":"Skøyen Stasjon",
        "address":"Skøyen Stasjon",
        "lat":59.9226729,
        "lon":10.6788129,
        "capacity":20
}
```


Figure 13. An example JSON object standing for the response of the station information endpoint of the API of the City Bike BSS in Oslo

Last but not least, the response from the call to get information about bike terminal availabilities contains an array of JSON objects again, each one standing for a bike terminal of the BSS. A JSON element in the same array within this response is like the one below.


```
    {
        "is_installed": 1,
        "is_renting": 1,
        "num_bikes_available": 7,
        "num_docks_available": 5,
        "last_reported": 1540219230,
        "is_returning": 1,
        "station_id": "175"
    }
```


Figure 14. An example JSON object standing for the response of the station availability endpoint of the API of the City Bike BSS in Oslo

The most important data take place in the response from the call to bike terminal availability history. But regarding the bike terminal, in a JSON object within the response, the only information is `station_id` which stands for bike terminal ID. This is actually a reference key. From the response from the call to the bike terminal information dataset, selecting by `station_id` value, referred bike terminal information (name, address, coordinates, capacity) can be reached. Embracing this approach, in this study, the eventual dataset for the City Bike BSS in Oslo was constructed by merging two responses.

In a way similar to the Vélib' BSS in Paris, the raw dataset builder for the City Bike in Oslo was executed by a job scheduler with an interval of 30 minutes for a limited amount of time. Eventually, in the CSV file, there were more than 100,000 records accumulated. This constructed raw dataset was then preprocessed and normalized to get ready to be processed by the prediction model.

3.3 Data Description

Parsing and deserializing API responses, eventually, there are two datasets having been written down into dedicated CSVs, one being for the Vélib' BSS in Paris and the other one being for the City Bike BSS in Oslo. There is another dataset too, as mentioned, which covers the availability history of a bike terminal in Suzhou. But it was already provided by the authors of the relevant paper. It includes more than 45,000 records. On the other hand, it should be noted that these datasets contain raw data which should be preprocessed and normalized to get ready to be processed by the prediction model, if necessary.

The raw dataset for the Vélib' BSS is composed of the fields described in the table below.

Field	**Description**	**Example Value**
dataset_id	Dataset ID	velib-disponibilite-en-temps-reel

record_id	Record UUID	1570b7d512f1fd3d40eafb63631c6ddd8d523b43
ebike	The number of electrical bikes at a terminal	8
capacity	The capacity of a terminal	60
name	The name of a terminal	Charonne – Robert et Sonia Delauney
nom_arrondissement_communes	The name of the region a terminal fall into	Paris
num_bikes_available	The number of available bikes at a terminal	15
is_installed	Whether bike terminal is installed or not	OUI It means “Yes” in French. NON It means “No” in French.
is_renting	Whether bike terminal is available for rent or not	OUI It means “Yes” in French. NON It means “No” in French.
mechanical	The number of mechanical bikes at a terminal	20
station_code	The bike terminal ID	10013

num_docks_available	The number of available docking units at a terminal	43
due_date	The timestamp of the due date of a bike terminal	2022-01-02T21:30:06+00:00
is_returning	Whether rented bikes shall be returned to the bike terminal or not	OUI It means “Yes” in French. NON It means “No” in French.
geometry_type	The type of the coordinates of a bike terminal	Point
longitude	The longitude of a bike terminal	2.3061046109
latitude	The latitude of a bike terminal	48.871044052
record_timestamp	The timestamp when the historical record was added to the database	2022-01-02T21:45:00.69400+00:00

Table 4. The description of the raw dataset for the Vélib' BSS in Paris

The raw dataset for a single bike terminal in Suzhou is relatively simple. Moreover, it includes fewer data since it only covers the history of a single bike terminal.

Field	**Description**	**Example Value**
num	Number of available bikes	5
weekday	The ordinal number of the weekday	4 (Thursday)

hour	The ordinal number of the hour when the record is added to the dataset	18

Table 5. The description of the raw dataset for a bike terminal in Suzhou

The built raw dataset for the City Bike BSS in Oslo is composed of the fields described in the table below.

Field	**Description**	**Example Value**
station_id	The unique ID of a bike terminal	175
station_name	The name of a bike terminal	Skøyen Stasjon
station_address	The address of a bike terminal	7 Juni Plassen
station_lat	The latitude of a bike terminal	59.9150596
station_lon	The longitude of a bike terminal	10.7312715
station_capacity	The capacity of a bike terminal	15
is_installed	Whether a bike terminal is installed or not	1 It corresponds to true. 0 It corresponds to false.
is_renting	Whether a bike terminal is available for renting or not	1 It corresponds to true. 0 It corresponds to false.

num_bikes_available	The number of available bikes at a bike terminal	7
num_docks_available	The number of available docking units at a bike terminal	5
is_returning	Whether rented bikes shall be returned to the bike terminal or not	1 It corresponds to true. 0 It corresponds to false.
last_reported	The timestamp when the historical record was added to the database	1540219230 In seconds

Table 6. The description of the raw dataset for the City Bike BSS in Oslo

3.4 Feature Engineering on Raw Datasets

It is required to preprocess and normalize the raw datasets having been constructed if needed to extract data suitable to be processed well by an ML/DL prediction model as much as possible. This operation is called feature engineering. According to Patel, ML/DL involves feature engineering approaches as an essential issue. Basically, the sequence of the processes of choosing, managing, and converting raw data into features suitable for supervised learning is called feature engineering. The primary purpose of doing it is to extract better measurable inputs, i.e. features, to be trained by a prediction model [24].

The relatively simple raw dataset for Suzhou did not require any feature engineering process in this study. Therefore, it was used as-is. But feature engineering was carried out for the raw datasets for the Vélib' BSS in Paris and the City Bike BSS in Oslo.

First of all, it must be taken into consideration that the target feature in the datasets that are supposed to be predicted by the developed prediction model should be bike availability ratio rather than the number of available bikes or docks. Therefore, within the context of feature engineering, for the Vélib' BSS in Paris and the City Bike BSS in Oslo, a new column called "*bike_availability_ratio*" was added, and the "*capacity*" column

from the dataset for the Vélib' BSS in Paris and "*station_capacity*" column from the dataset for the City Bike BSS in Oslo were removed. The newly added column was filled with the percentages of available bikes, and they were calculated by Equation 3.1 below.

$$bike\ availability\ ratio = 100 * \frac{number\ of\ available\ bikes}{capacity} \tag{3-1}$$

The second step was the normalization of geolocation. Known that the longitude and the latitude of a point cannot be greater than 100 or less than 0, simply by dividing them to 100, they can be scaled down to the range [-1,1]. Geolocation information in both datasets was normalized in that way. For example, a geolocation coordinate (2.3522, 48.8566) could be transformed into (0.023522, 0.488566) through the aforementioned normalization.

The third step was date and time conversions. First and foremost, for the dataset of the City Bike BSS in Oslo, the *last_updated* column was converted into the regular timestamp format from seconds, and the conversion was saved into the new column named after *record_timestamp* as in the dataset for the Vélib' BSS in Paris. (The *last_updated* column was then removed as it became considered redundant for time being.) Within the context of this step, all timestamp values were divided into the smallest units and the extracted values were put into new columns as below.

- record_timestamp_year
- record_timestamp_month
- record_timestamp_day
- record_timestamp_hour
- record_timestamp_minute
- record_timestamp_second

Each column holds the ordinal number of the date/time unit it represents. For example, if the record timestamp covers the date 24-12-2021 and the time 15:11:07, upon the process

mentioned above, the values corresponding to the transformed timestamp will be as below.

- record_timestamp_year: 2021
- record_timestamp_month: 12
- record_timestamp_day: 24
- record_timestamp_hour: 15
- record_timestamp_minute: 11
- record_timestamp_second: 7

In the Python programming language, this extraction can be achieved by the *DateTimeIndex* method of the *pandas* library.

The next step was to remove redundant columns. Within the context of this step, all columns that were considered relatively unhandy in the perspective of the trainer prediction model were removed from the latest forms of both datasets that have been modified through the previous steps. As a result, from the dataset for the Vélib' BSS in Paris, the following columns were supposed to be essentially processed to make an estimation by the used models left.

- is_installed
- is_renting
- is_returning
- longitude (normalized)
- latitude (normalized)
- bike_availability_ratio (target)
- record_timestamp_year
- record_timestamp_day

- record_timestamp_hour
- record_timestamp_minute
- record_timestamp_second

The final step of feature engineering was label encoding. As mentioned before, the columns whose names start with “is_” take either OUI or NON which means “Yes” and “No” respectively in French. Instead of letting them be processed with these values, they were encoded in the following way.

- OUI = 1
- NON = 0

In the Python programming language, it can be achieved by calling the *fit_transform* function of the *sklearn* library.

Within the scope of the same step, record timestamp index columns were also transformed into dummies. For each unique index (ordinal) date/time unit value in these columns, a new column was created. For a record, columns that hit the timestamp of the record were filled with 1 whereas the rest of them were with 0. For example, for the timestamp 26-09-2021 11:04:15, the following columns will be created if they have not been created because of another timestamp in the dataset yet.

- record_timestamp_year_2021
- record_timestamp_month_9
- record_timestamp_day_26
- record_timestamp_hour_11
- record_timestamp_minute_4
- record_timestamp_second_15

For the relevant row, these columns will be filled with 1. Other columns for the same row will be filled with 0. For example, for the same row, a column named after

record_timestamp_day_18, which might have been created because of another timestamp with the day value of 18 in the dataset, would be filled with 0 as it does not hit the day of the timestamp of the record belonging to the row.

In the Python programming language, this normalization can be achieved by calling the *get_dummies* function from the *pandas* library. It also removes the formerly processed "record_timestamp_" columns as they are redundant from then on.

Ultimately, the preprocessed and normalized form of the dataset for the Vélib' BSS in Paris consists of the fields described below.

Field	**Description**	**Example Value**
is_installed	Whether the bike terminal is installed or not	1 or 0
is_renting	Whether the bikes are available to be rented or not	1 or 0
is_returning	Whether the bikes shall be returned to the belonging bike terminal or not	1 or 0
longitude	The longitude of a bike terminal falling into the range [-1,1]	0.02367
latitude	The latitude of a bike terminal falling into the range [-1,1]	0.48753
bike_availability_ratio	The percentage of available bikes in the capacity of the bike terminal	22.580645 (%)

record_timestamp_year_n	Whether the timestamp of the record has the year value n or not	1 or 0
record_timestamp_month_n	Whether the timestamp of the record has the month value n or not	1 or 0
record_timestamp_day_n	Whether the timestamp of the record has the day value n or not	1 or 0
record_timestamp_hour_n	Whether the timestamp of the record has the hour value n or not	1 or 0
record_timestamp_minute_n	Whether the timestamp of the record has the minute value n or not	1 or 0
record_timestamp_second_n	Whether the timestamp of the record has the second value n or not	1 or 0

Table 7. The columns of the preprocessed dataset for the Vélib' BSS in Paris

The final form of the dataset on which feature engineering was done was written down into another CSV file, as can be seen from Fig 10.

Figure 15. The CSV file of the preprocessed dataset for the Vélib' BSS in Paris opened with Visual Studio Code

When it comes to the dataset for the City Bike BSS in Oslo, similar processes were treated using the same techniques and libraries. The conclusive form of the preprocessed and normalized dataset for the City Bike BSS in Oslo as a result of feature engineering consists of the fields described below.

Field	**Description**	**Example Value**
is_installed	Whether the bike terminal is installed or not	1 or 0
is_renting	Whether the bikes are available to be rented or not	1 or 0
is_returning	Whether the bikes shall be returned to the belonging bike terminal or not	1 or 0
station_longitude	The longitude of a bike terminal falling into the range [-1,1]	0.02367
station_latitude	The latitude of a bike terminal falling into the range [-1,1]	0.48753
bike_availability_ratio	The percentage of available bikes in the capacity of the bike terminal	22.580645 (%)
record_timestamp_year_n	Whether the timestamp of the record has the year value n or not	1 or 0

record_timestamp_month_n	Whether the timestamp of the record has the month value n or not	1 or 0
record_timestamp_day_n	Whether the timestamp of the record has the day value n or not	1 or 0
record_timestamp_hour_n	Whether the timestamp of the record has the hour value n or not	1 or 0
record_timestamp_minute_n	Whether the timestamp of the record has the minute value n or not	1 or 0
record_timestamp_second_n	Whether the timestamp of the record has the second value n or not	1 or 0

Table 8. The columns of the preprocessed dataset for the City Bike BSS in Oslo

The final form of this dataset was written down into a separate CSV as well.

4 Prediction Model

Constructed preprocessed and normalized datasets are ready to be trained by ML/DL models to estimate bike terminal availabilities. The prediction engine component of the proposed architecture was supposed to put the prediction model developed here to the centre of it. Therefore, working on a prediction model was one of the essential phases of the research here.

An approach that both outperforms state of art and some other commonly used techniques used in this research by the aspect of accuracy and results in satisfactory results was expected to exist. It was thought that an existing approach could be considered baseline and it could be tuned up to boost the accuracy benchmark.

Preparation for training involves dataset separation as a crucial step. A dataset is generally divided into a training set and a testing set. A third partition called validation set may also be preferred. Shah implies that a model is fitted by the training partition, which means that the model observes and analyses the training data and learns the pattern from it, whereas the testing partition provides an objective evaluation for the model. In addition, the training partition is usually far bigger than the testing partition, such as 70% of a dataset can be considered training data and the remaining 30% can be used as testing data [25]. Hereby, overfitting is also another issue that noteworthily impacts the accuracy of a model and is related to the training-testing data separation. According to Ying, overfitting is a situation the pattern learning performance of a model is notably poor since it does not fit on testing data at a satisfactory level whereas it fits on the training data very well [26]. During training, it should either not occur at all or occur late and at a tolerable level. Therefore, another substantial point that was taken into consideration was to mitigate the risk of overfitting as much as possible.

Within the context of this research, taking literature review and state of the art into account as well, a Recurrent Neural Network (RNN) model Long Short-Term Memory (LSTM) was chosen to work on. For the sake of comparison, apart from techniques where LSTM is abstracted, generated processed datasets were trained by Random Forest Regressor and Decision Tree Regressor models as well. Then the results of these training were compared to each other by their mean absolute error (MAE) and mean squared error

(MSE) scores. Briefly, the lower MAE and MSE a training yields the better performance it has.

For an ML/DL model, mean squared error is the ratio of the sum of the square of each difference between prediction and truth to the entire dataset size [27]. It can be calculated by the following equation, where *N* is the number of data, *y* is truth, and *x* is prediction.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)\ \hat{}\ 2$$

(4-1)

For an ML/DL model, mean absolute error differs from MSE in terms of the things being summed up. Here, the sum of the absolute value of each difference between prediction and truth is divided by the entire dataset size [27]. It can be calculated by the following equation where variables have the same meaning.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_i - x_i|$$

(4-2)

4.1 Long Short-Term Memory

Before addressing the details of the researched tune-up, the details of LSTM and why it was chosen to be abstracted for this research should be clarified.

It is indicated by Mushailov that LSTM (Long Short-Term Memory) is a model based on Recurrent Neural Network (RNN), and time series prediction and natural language processing are two of the commonly used fields. The thing is short-memory cannot be handled well by recurrent neural networks. LSTM solves this remarkable issue through gates in memory cells. Within the context of this architecture, data can be kept, forgotten, or ignored. It works based on a feedback mechanism. Once estimation is output, the prediction is fed back into the system for the estimation of the upcoming value in the sequence. Another important issue that can be solved by LSTM is that in each epoch, which is a unit time for each feedback, the architecture receives some error information

as well. In order to prohibit the explosion and vanishing of repetitive weight adjustments, which can occur in other neural network models, sigmoid and tanh activation functions are carried out before entering into the next gate and leaving the current gate. These functions help the architecture squeeze gradients that could explode [28].

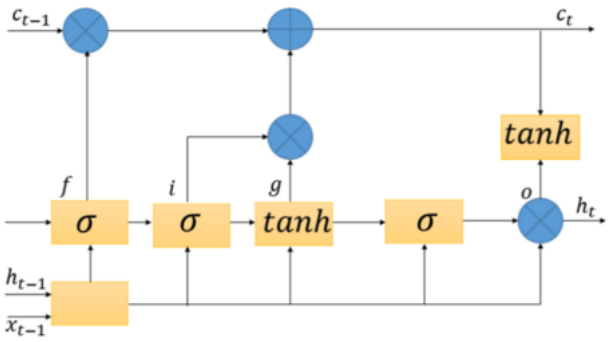

Figure 16. An LSTM cell with gates that helps data be processed sequentially

Figure 16 demonstrates the architecture of an LSTM memory cell with gates [29]. It demonstrates the flow between gates. Here, x_{t-1} is the new input, h_{t-1} is the input from the previous cell corresponding to the previous time value, c_{t-1} is the previous cell state. *f*, *i*, and *g* are respectively forgetting, input, and output gates. "x" and "+" denote operations (addition or multiplication) carried out.

Since datasets being processed in this research are composed of time series data, which means rows are ordered by timestamp in a sequence and each row has information about bike availability at its timestamp, LSTM was considered a suitable architecture. It can be inferred that each row in a dataset is handled by a memory cell of the implemented LSTM model. On the other hand, considering the fact that datasets are huge, and they can rapidly grow gradually as long as the input to the system is provided, the prediction model must observe data over a long time period. LSTM outcomes satisfactory results with a huge amount of time series data [29]. That is why LSTM was chosen to be abstracted in this research.

4.2 Developed Approach

A basic LSTM implementation like the one in the research [10] was tuned up to outcome better accuracy with less error and late and minimum outfitting. To achieve this, the

parameters and layers of the basic LSTM implementation were modified, and each modification was tested with training to check if the results are better than those of the previous training.

First of all, Keras and Tensorflow libraries of Python programming language were used for building the architecture since these libraries already provide the necessary functions for RNN.

From a trained dataset, 30% of data was used for testing, and the remaining 70% was used for training.

The basic LSTM implementation did not include the bidirectionality layer of Keras. This layer was added to the development. This layer makes the system eligible to observe data both forward and backwards in the same sequence. Otherwise, by default, the mechanism is run only forward. The architecture can be trained better in this way because of learning more patterns.

The "*units*" parameter of the LSTM architecture was ascended to 128 from 50 which was used in the basic implementation as in the study [10]. This parameter adjusts the dimension number of the output space [30]. It is used to define how many dimensions the output will be represented with. Even though increasing it affects the training runtime duration, the more units an LSTM model has the better accuracy is possible to approach.

Both LSTM abstracted solutions were trained several times to output information about each epoch. For each epoch, the metrics mean absolute error and mean squared error were calculated and the loss was expected and observed to be less than that in the previous epoch. In the developed approach, a mechanism called "Early Stopping" was added to mitigate the overfitting problem. It is a function of the Keras library [31]. Basically, if a metric passed as a parameter to the early stopping function, e.g. mean absolute error, starts to fluctuate between two values and accordingly the training stops developing apparently, the aforementioned function stops the training process to prevent a possible overfitting appearance. For example, when the developed model was trained with the dataset for the Vélib' BSS in Paris, the training stopped after 4 epochs, because MSE values were about to fluctuate. Possible overfitting was also prohibited in this way. However, it was supposed to run over 20 epochs. The former model with the basic LSTM implementation completed all 20 epochs as it was not instructed to stop when needed.

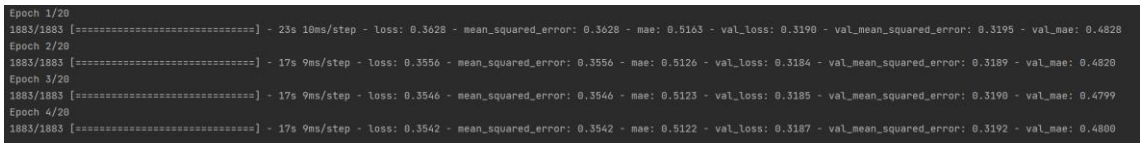

Figure 17. All passed 4 epochs with the early stopping mechanism

For both models, epoch numbers were set to be 20, batch sizes were set to be 32, and the dropout rates were set to be 0.5.

4.3 Evaluation of the Predictive Performance

To evaluate the predictive performances of all models and compare them to each other, the first metric to be considered was mean absolute error in uniform average.

Dataset	Basic LSTM	Random Forest Regressor	Decision Tree Regressor	Developed approach
Vélib' BSS in Paris	0.532991	2.200753	5.334560	0.511285
Suzhou	0.649578	4.315326	4.295474	0.118561

Table 9. Mean absolute error (MAE) comparison

From the comparison result, it was inferred that the tuned-up LSTM-based model outperformed the basic LSTM implementation, which was also in the research [10], as well as Random Forest Regressor and Decision Tree Regressor in terms of mean absolute error, since the developed approach output relatively low MAE. On the other hand, the acquired MAE values could be considered as low as satisfying to rely on the predictions it would make.

The second evaluation was done by comparing mean squared error differences between evaluation for training data and evaluation for testing data both for the basic LSTM implementation and the tuned-up LSTM-based approach. This is also related to how well the overfitting problem was solved. The dataset for the Vélib' BSS in Paris was used for all training.

```
np.abs(model.evaluate(X_train, y_train)[1]-model.evaluate(X_test, y_test)[1])

 2092/2092 [==============================] - 3s 1ms/step - loss: 0.3430 - mean_squared_error: 0.3430 - mae: 0.5013
 897/897 [==============================] - 2s 2ms/step - loss: 0.3711 - mean_squared_error: 0.3712 - mae: 0.5330

 0.028165310621261597
```

Figure 18. Evaluation for training data and evaluation for testing data results for the basic LSTM implementation

```
np.abs(model.evaluate(X_train, y_train)[1]-model.evaluate(X_test, y_test)[1])

 2092/2092 [==============================] - 4s 2ms/step - loss: 0.3501 - mean_squared_error: 0.3501 - mae: 0.5082
 897/897 [==============================] - 2s 2ms/step - loss: 0.3721 - mean_squared_error: 0.3722 - mae: 0.5313

 0.02210026979446411
```

Figure 19. Evaluation for training data and evaluation for testing data results for the developed LSTM-based model

As can be seen from the Jupyter Notebook particles in Figure 18 and Figure 19, MSE differences written down at the bottom are different. The value for the developed LSTM-based approach was 0.022100 whereas the one for the basic LSTM implementation was 0.028165. In the light of this information, it can be inferred that the developed LSTM-based model outperformed the basic LSTM model and gave an affirmative potential to mitigate the existence of overfitting between training data and testing data.

5 Development Environment

Dataset building, dataset processing, model development and training phases constitute the development process within the scope of the study in this paper. The development was done on two computers with the specifications as below. The software and hardware configurations of a computer affect the accuracy and runtime performance of an ML/DL model as well.

- Windows-based PC
    - Windows 11 Home (x64)
    - 9th Generation Intel Core i7 CPU
    - 16 GB of installed RAM
    - NVIDIA GeForce RTX 2080 GPU
    - 512 GB of SSD
- MacBook Air (Retina, 13.3”, 2019)
    - macOS Monterey 12.2
    - 8th Generation Intel Core i5 CPU
    - 8 GB of installed RAM
    - Intel UHD Graphics 617 GPU
    - 256 GB of SSD

For the development and reporting processes, the following software development and documentation technologies on both computers were used.

- Python 3.8 programming language with necessary libraries including Keras, TensorFlow, scikit-learn that are ML/DL oriented
- PyCharm IDE

- Jupyter Notebook
- Microsoft Visual Studio Code
- Microsoft Word
- Microsoft Excel

6 Proposed Software Architecture

The study includes a proposal for the architecture of an extensional application into which the prediction engine based on the developed approach can be integrated too. A dashboard where a map that points out bike terminals around the current location of end-user takes place, these bike terminals can be filtered, and the most available bike terminals are declared to the user was proposed to be designed and put into a mobile application or a website. The most important concept here is the architecture on the backend side of the dashboard. This architecture is presented in Figure 20.

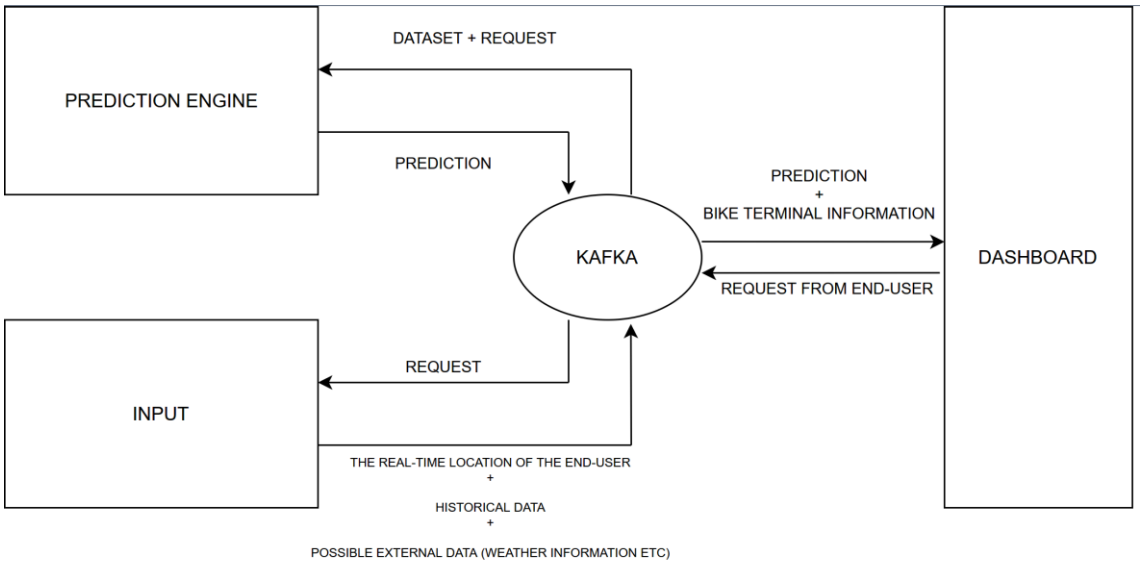

Figure 20. The proposed architecture for the application backend

The architecture was proposed to be composed of four major parts: input, prediction engine, Kafka, and dashboard. Each one can be considered a sort of microservice or module. The Kafka part of the architecture was proposed to run Apache Kafka framework to handle message transfer between modules [32]. Here, a message can be a simple request or data such as prediction result, user location, or historical data. The input module of the architecture was proposed to provide any kind of input data (user location, historical data, external data such as weather or traffic density) to the system if there is a user request coming from the dashboard module of the architecture. The provided input is supposed to be sent to the prediction engine module that trains data using the developed estimation algorithm. This module was proposed to make estimations about bike terminal

availabilities. The predictions were proposed to be sent to the dashboard module eventually.

7 Conclusion and Future Work

In this paper, the main purpose is to seek a new outperforming solution for the problem of predicting bike terminal availability. The paper starts with addressing the main motivation to conduct research over the problem and the problem definition itself. Then background information that covers a literature review to see the state of the art regarding the same problem takes place. According to the literature review, it can be inferred that there are several studies regarding the estimation about the occupancy of a bike terminal or multiple bike terminals of a bike-sharing system. Depending on data type and quality too, manifold machine learning and deep learning techniques result in dissimilar accuracies that can be measured by several metrics such as mean absolute error or mean squared error. The data and prediction model chapters cover the actual implementation phase within the context of the study here. To train the developed solution, datasets from the Vélib BSS in Paris, France, the City Bike BSS in Oslo, Norway, and a single bike terminal in Suzhou, China were built. Since they contain time-series data, a new approach that could be formed around Long Short-Term Solution (LSTM) Recurrent Neural Network (RNN) was considered the most suitable solution. Basically, by tuning up the architecture of a basic LSTM implementation, a new model that outperforms the basic LSTM implementation and some other machine learning techniques with lower mean squared error and mean absolute error scores was developed. Ultimately, this paper proves that it is possible to obtain enhanced accuracy, accordingly better predictions about bike terminal availabilities, by tuning up the parameters and the architecture of an existing technique.

There are several possible works that can be considered in the future. Since the prediction engine module is proposed to be dynamic, several datasets from different cities around the world can be used for training as long as they are accessible. Data scraping for Tallinn, Estonia can be proposed if there are available data sources. The application whose architecture proposal is explained in the chapter “6 Proposed Software Architecture” can be developed and put into practice. It is possible to conduct further researches about possible solutions to improve accuracy, including tuning up an existing basic ML/DL model. Moreover, in this paper, external data such as historical weather conditions or traffic density was not integrated. The impact of their integration into the datasets shall be examined.

References

[1] A. Bernhard, “The great bicycle boom of 2020,” BBC, [Online]. Available: https://www.bbc.com/future/bespoke/made-on-earth/the-great-bicycle-boom-of-2020.html.

[2] M. M. Hasan, U. Attanayake and L. Lopez, “Infrastructure and technology for sustainable livable cities,” Michigan, 2016.

[3] “About Blue Bikes Boston,” Bluebikes, [Online]. Available: https://www.bluebikes.com/about.

[4] C. McDonnell, “To bike or not to bike? Machine learning to model availability in San Francisco's bike share program,” Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/sf-bike-share-predictions-e15316ce300f.

[5] E. Glusac, “Farther, faster and no sweat: bike sharing and the e-bike boom,” The New York Times, 2021. [Online]. Available: https://www.nytimes.com/2021/03/02/travel/ebikes-bike-sharing-us.html.

[6] S. Brown, “Machine learning, explained,” MIT Management Sloan School, 2021. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained.

[7] “What is deep learning,” Oracle, [Online]. Available: https://www.oracle.com/data-science/machine-learning/what-is-deep-learning/.

[8] C. Xu, J. Ji and P. Liu, “The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets,” *Transportation Research Part C: Emerging Technologies,* vol. 95, pp. 47-60, 2018.

[9] H. I. Ashqar, M. Elhenawy, H. A. Rakha, M. Almannaa and L. House, “Network and station-level bike-sharing system prediction: a San Francisco bay area case study,” *Journal of Intelligent Transportation Systems,* 2021.

[10] X. Liu, A. Gherbi, W. Li and M. Cheriet, “Multi features and multi-time steps LSTM based methodology for bike sharing availability prediction,” *Procedia Computer Science,* vol. 155, pp. 394-401, 2019.

[11] W. Zi, W. Xiong, H. Chen and L. Chen, “TAGCN: station-level demand prediction for bike-sharing system via a tempoeral attention graph convolution network,” *Information Sciences,* vol. 561, pp. 274-285, 2021.

[12] D. Johnson, “What is a CSV file? How to open, use, and save,” Business Insider, 2021. [Online]. Available: https://www.businessinsider.com/what-is-csv-file.

[13] “About Vélib' - Vélib' metropole,” Vélib' Metropole, [Online]. Available: https://www.velib-metropole.fr/en/service.

[14] M. Jacobson, “Paris Vélib' bike-sharing,” WWF, 2012. [Online]. Available: https://wwf.panda.org/wwf_news/?204579/Paris-Vlib-bike-sharing.

[15] F. Hernandez, “Paris: Pendant trois mois, Vélib' fait moins 50% sur les abonnements,” 20 Minutes, 2018. [Online]. Available: https://www.20minutes.fr/paris/2324591-20180823-paris-pendant-trois-mois-velib-fait-moins-50-abonnements.

[16] “About Oslo city bike,” Oslo City Bike, [Online]. Available: https://oslobysykkel.no/en/about.

[17] “City bikes: Oslo bysykkel,” Visit Norway, 2021. [Online]. Available: https://www.visitnorway.com/listings/city-bikes%3A-oslo-bysykkel/550/.

[18] “Paris Data,” [Online]. Available: https://opendata.paris.fr/pages/home/.

[19] “Licence de réutilisation des données,” Paris Data, [Online]. Available: https://opendata.paris.fr/page/lalicence/.

[20] “Search API v1 Documentation,” Opendatasoft, [Online]. Available: https://help.opendatasoft.com/apis/ods-search-v1/#search-api-v1.

[21] “Realtime data,” Oslo City Bike, [Online]. Available: https://oslobysykkel.no/en/open-data/realtime.

[22] “Norsk lisens for offentlige data (NLOD) 2.0,” Felles datakatalog, [Online]. Available: https://data.norge.no/nlod/no/2.0/.

[23] “General Bikeshare Feed Specification (GBFS),” NABSA, 2015. [Online]. Available: https://github.com/NABSA/gbfs/blob/master/gbfs.md.

[24] H. Patel, “What is feature engineering - importance, tools, and techniques for machine learning,” Towards Data Science, 2021. [Online]. Available: https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10.

[25] T. Shah, “About train, validation, and test sets in machine learning,” Towards Data Science, 2017. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

[26] X. Ying, “An overview of overfitting and its solutions,” *Journal of Physics,* 2019.

[27] G. Seif, “Understanding the 3 most common loss functions for machine learning regression,” Towards Data Science, 2019. [Online]. Available: https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3.

[28] J. Mushailov, “LSTM framework for univariate time-series prediction,” Towards Data Science, 2021. [Online]. Available: https://towardsdatascience.com/lstm-framework-for-univariate-time-series-prediction-d9e7252699e.

[29] M. Elsaraiti and A. Merabet, “A comparative analysis of the ARIMA and LSTM predictive models and their effectiveness for predicting wind speed,” *Energies,* pp. 1-16, 2021.

[30] “LSTM layer,” Keras, [Online]. Available: https://keras.io/api/layers/recurrent_layers/lstm/.

[31] “Early stopping,” Keras, [Online]. Available: https://keras.io/api/callbacks/early_stopping/.

[32] “Kafka,” Apache, [Online]. Available: https://kafka.apache.org/.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Teoman Turan

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Real-Time Prediction for Bike Terminal Availability” supervised by Sadok Ben Yahia, Wissem Inoubli
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

03.01.2022

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – GitHub Repository for Source Code

The work was committed-and-pushed to the following GitHub repository where the source code of the project can be found:

https://github.com/tetura/RealtimePredictionBikeTerminalAvailability