An AutoML Approach for Bike Demand Forecasting and Redistribution

Dimitris Petratos[1], Yannis Poulakis[1], Irene Gimenez Pedralba[2], Cristina Aragon Garcia[2], and Christos Doulkeridis[1(✉)]

[1] Department of Digital Systems, University of Piraeus, Piraeus, Greece
{dpetr,gpoul,cdoulk}@unipi.gr
[2] Serveo, Barcelona, Spain
{igimenez,cristina.aragon}@serveo.com

Abstract. In this paper we introduce a two-staged pipeline to tackle the problem of bike redistribution for bike-sharing systems, using Automated Machine Learning (AutoML) and optimization techniques. Our approach includes the usage of AutoML for time series forecasting in order to estimate the demand for bikes for each station, along with an optimization model to efficiently relocate bikes to maximize user satisfaction. In our study, we used historical data from Barcelona's public bike-sharing system to predict future demand and then used these predictions together with public data from OpenStreetMap (estimated travel time between stations) in order to solve the Minimum Cost Flow Problem (MCFP) and compute the optimal bike redistribution. We demonstrate promising results in terms of accuracy of demand forecasting and reduction of forecasting time, thus obtaining feasible redistribution strategies and providing an end-to-end framework to the operator.

Keywords: Bike demand forecasting · time series forecasting · AutoML · bike redistribution · minimum-cost flow problem

1 Introduction

During the last decades due to high urbanization rates, large cities face several challenges in planning and specifically in terms of transportation. The growth of cities' population has positioned citizens' transportation as one of the most complex difficulties that policy makers have to overcome. At the same time the ever increasing pollution from car use, calls for the immediate adoption of eco-friendly approaches that minimize the energy footprint of transport. Such solutions include bike-sharing systems (BSSs) that the citizens can use instead of traditional transportation means such as cars, buses etc. BSSs are highly adopted from large crowded cities all over the globe and can efficiently satisfy citizens' transportation needs in a green manner [5]. BSSs can be divided into two categories. If the bikes are picked up and dropped off at stations in specific

A. Kocian et al. (Eds.): INTSYS 2024, LNICST 608, pp. 145–161, 2025.
https://doi.org/10.1007/978-3-031-86370-7_9

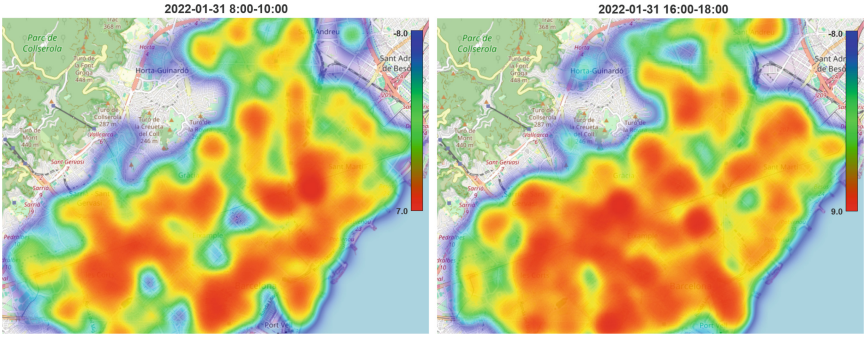

Fig. 1. Bike demand density (of stations) for two different time intervals, 8:00–10:00 and 16:00–18:00 for 31/1/2022. The two intervals illustrate different user behavior. This observation motivates the need for a dynamic redistribution strategy.

geographic locations, the systems are known as *Station-Based* (SB-BSSs). Otherwise, if the users can end their ride and leave the bike at any point to be picked up again for another trip, the systems are known as *Free-Floating* (FF-BSSs).

These systems are greatly affected by the dynamic nature of user behavior. Bike trips are tightly connected to spatio-temporal information, i.e., users are more likely to pick up bikes in urban areas early in the day and drop them off in the same area later in the day. Additionally, bike trips may be affected by events like extreme weather conditions or festivals. This creates imbalances in supply and demand which affects user satisfaction and has economic impact on the organization operating the BSS. To tackle this challenge, BSSs usually include a redistribution strategy to relocate idle bikes where demand is expected to be high. Such strategies can be classified as static and dynamic, namely *Static* (S-BSS) and *Dynamic* (D-BSS), respectively. In the static case, bikes are redistributed at a fixed time in the day, usually at some point in the night when demand is at its lowest, to even out the bike spread over the region for the next day. Contrary to that, in the dynamic approach the redistribution may happen at several points in time in a day, usually at fixed time intervals. While static redistribution has less operational costs associated with, it is incapable of reacting to dynamic changes in bike demand. In Fig. 1 we illustrate the need of a dynamic redistribution approach based on the dataset we used for this paper.

Whatever the BSS operational type, accurate demand prediction is of the essence. Without modeling and predicting the bike demand for a future time period, however good may a redistribution model be, it will fall short as it will fail to meet the actual user activity. In BSSs it is especially difficult, because users do not go through a vehicle request process, which usually takes place in other vehicle systems. Historically, this demand prediction has been of active research as it is necessary for the redistribution process. Motivated by this, in this paper, we address the problem of combined demand prediction and redistribution planning using Automated Machine Learning techniques and in particular

Bayesian Optimization (BO) to tackle the problem of demand prediction, along with redistribution planning based on a linear programming problem defined in the next sections, in order to provide operators with an optimal redistribution strategy.

In summary, we make the following contributions in this paper:

- We cast the problem of demand forecasting for bike-sharing systems as an AutoML-based time series forecasting problem, in order to obtain highly accurate predictions of bike demand.
- We formalize the problem of bike redistribution among the stations of a bike-sharing system based on the Minimum Cost Flow (MCF) problem.
- We present experiments using real-world data from a bike rental operator (Serveo) in Barcelona to evaluate the effectiveness of our approach.

The rest of the paper is structured as follows. Section 2 provides an overview of related work. Section 3 formally defines the problem. Then, in Sect. 4, we present our pipeline for bike demand forecasting and bike redistribution. Section 5 reports the results of the experimental evaluation. Finally, Sect. 6 concludes the paper and points to future research directions.

2 Related Work

The vehicle redistribution problem has been researched under various contexts. One specific area of study is that of empty vehicle redistribution, inspired by the recent traction of autonomous driving vehicles. Some of the recent works [2,10] that focus on this sub-domain, are modeled to include knowledge on passenger wait-time which they model according to proposed utility functions.

Bike-Sharing Systems. Bike redistribution is also being actively researched. In [3,11], a set of mathematical formulations for bike demand and the rebalancing problem are presented. Additionally possible constraints that may be taken into account under different applications are presented. Finally, in the lastly aforementioned work, proposed algorithms and methodologies are benchmarked.

Station clustering has been explored in [6,17]. In [17], the authors use a community detection approach on the graph that represents bike stations (vertices) and bike flows (directed edges). In [6], the bike stations are first classified as supply and demand stations and then are clustered accordingly to create a super-node per cluster. The station matching for bike redistribution is performed upon these produced nodes.

In [12], total station demand is calculated by separately predicting pick up rate and drop off rate for 1h time intervals. The pick up rate is predicted through a weighted KNN that takes into account weather information as well. The drop off rate is predicted based on predicted pick up rate by leveraging a multimodal gaussian distribution to estimate whether picked up bicycles will arrive at their destination at this interval or the next. Rebalancing afterwards is achieved through Mixed Integer Programming on clustered stations to eliminate outliers.

In [15], the authors formulate the BRP in four stages: (1) a demand forecasting model that anticipates inventory levels at different time intervals, (2) station inventory model determines the optimal initial inventory level such that in the next time period system inefficiencies, (3) redistribution needs model that models the redistribution's plan and (4) vehicle routing model that maximizes the utility of redistribution vehicles. Both (1) and (2) are taking advantage of heterogeneous data such as inventory level, traffic and weather and are trained in a dynamic manner (periodically). For (1) the authors use gradient boosting machines and for (2), assuming that departures and arrivals at a station follow a Poisson distribution the authors compute their parameters from historical data. For (3) the authors formulate the redistribution problem as a stochastic linear integer program, whereas for (4) they use an arc and sequence-indexed formulation.

In this paper, we formulate the BRP using real-world historical data to compute the demand and combine this information with real-world traffic information to provide the operator with a redistribution strategy, solving a linear program problem.

AutoML for Time Series. Time series forecasting (TSF) methods can be broadly classified in statistical methods, machine learning methods and deep learning methods. The best performing method depends on various criteria; the application domain, the dataset at hand, the size of historical data, etc. Moreover, each method uses a large number of hyperparameters and finding the optimal values is a time-consuming task, both for the data scientist as well as in relation with the computational resources for evaluation of different setups. To cope with these challenges, a recent trend is to apply Automated Machine Learning (AutoML) [8] techniques in order to automatically detect the best forecasting algorithm and configuration of hyperparameters, without exhaustively testing a huge number of configurations. In the case of BRP where bikes' demand is represented by time series and results' provision needs to be short-termed, AutoML to deal with TSF is essential in order to provide redistribution operators with valid demand predictions very quickly.

AutoARIMA and AutoETS [9] are processes for automating parameter selection for the statistical algorithms ARIMA (Auto-Regressive Intergrated Moving Average) and EST (Exponential Smoothing). Hyndman and Khandakar included these methods for the first time in 2008 in an R package and they are based on the minimization of an information criterion like Akaike's Information Criterion (AIC) or the Bayesian Information Criterion (BIC).

AutoGluon-Time Series (AG-TS) [16] is a Python AutoML library for probabilistic time series forecasting. It contains a variety of forecasting algorithms: statistical, machine learning based and ensembles. It is easy to use and provides to the non-expert end user the ability to perform precise forecasts using a convex combination of forecasted values produced with different forecasters. In this way, AG-TS takes advantage of several predictors.

Auto-Pytorch-TS [4] is an AutoTSF framework for Automated Deep Learning time series forecasting, which was built as an extension of the Auto-Pytorch

[18] package for tabular datasets. The authors introduce a novel Neural Architecture Search (NAS) approach that deals with the joint problem of neural architecture search and data processing hyperparameters' optimization for time series data, taking advantage of BO.

In this paper, we address the AutoML for time series forecasting problem directly, using BO for a predefined number of iterations. In this way, not only we examine different configurations of the search space without human interaction to find the best algorithm's configuration, but we also provide the redistribution operator with the best forecasts in a fast manner.

3 Problem Definition

In this section, we present the problem setting for our work. We identify two subproblems, bike demand forecasting (Sect. 3.1) and bike redistribution (Sect. 3.2). Our approach addresses both of these subproblems, offering an end-to-end solution.

3.1 Bike Demand Forecasting

In this subsection we define the basic concepts of our problem that are connected to the bike demand forecasting problem. Let $\mathcal{S} = \{i : i = 1, \ldots, z\}$ denote a set of z stations of bikes, where a station is a fixed-place location of specific capacity (total number of bikes that can be placed there). Users can borrow bikes from a station or return a bike after the trip has been conducted. Supposing that we examine a time interval $[t_{start}, t_{end})$, let a uniform and non-overlapping discretization of n distinct time intervals $\mathcal{T} = \{T_1, \ldots, T_n\}$ where $T_i = [t_i, t_{i+1}) \subset [t_{start}, t_{end})$, so that $T_k \cap T_l = \emptyset,\ \forall T_k \neq T_l$ and

$$\delta\mathcal{T} = |T_i| = |t_{i+1} - t_i| = \frac{t_{end} - t_{start}}{n},\ \forall T_i \tag{1}$$

We define the demand of the i-th station to be the sequence (time series) $\mathcal{D}_i(t)$, that corresponds to the total number of bikes that departs or arrive at station i for a time interval that is denoted by the index t. The bike demand forecasting problem is that of predicting the demand $\mathcal{D}_i(T_j)$ for a station i at the time interval T_j.

3.2 The Bike Redistribution Problem

Here we formulate the so-called Minimum-Cost Flow Problem (MCFP) under the bike redistribution assumptions. Our goal is to redistribute bikes among stations in an optimal manner to cover future demand.

Let a flow network $\mathcal{G} = (V, E)$ be a directed graph with a source node $v \in V$ and a destination node $u \in V$ that represent stations. The source node v represents the station from which we take a number of bikes and redistribute them to the destination node u. Each edge $(v, u) \in E$ carries a *weight* $f(v, u)$

that represents the number of bikes that need to be transported from node v to node u Typically, the $f(v,u)$ weights are the problem's variables which we seek to estimate and they describe the number of bikes that operator has to transfer from station v to station u.

Additionally, each $(v,u) \in E$ has a *capacity* $cap(v,u)$, indicating the maximum number of bikes that the redistribution vehicles can transport. Also, each $(v,u) \in E$ is associated with $cost(v,u)$, a value which indicates the estimated travel time from station v to station u.

Finally, every $v \in V$ has a supply/demand indicator $d(v)$, which suggests whether a node v has surplus or shortage of bikes. More precisely, given $v \in V$ and for a fixed time interval t, $d(v)$ is equal to $\mathcal{D}_i(t)$ (as defined in Sect. 3.1) where v is the node that corresponds to station i.

The bike redistribution MCFP is formulated as a linear programming problem as follows:

$$\min_{\{f(v,u):(v,u)\in E\}} \sum_{(v,u)\in E} cost(v,u) \cdot f(v,u) \tag{2}$$

such that

1. $f(v,u) \leq cap(v,u),\ \forall (v,u) \in E$
2. $\sum_{w\in V} f(u,w) - \sum_{w\in V} f(w,u) = d(u),\ \forall u \in V$
3. $f(u,v) \geq 0,\ \forall (u,v) \in E$

The second constraint illustrates a notion of balance. This means that it enforces the difference of the outgoing flow $\left(\sum_{w\in V} f(u,w)\right)$ minus the incoming flow $\left(\sum_{w\in V} f(w,u)\right)$ of a node u, to be equal with its total demand $d(u)$ and is derived from the general assumptions of MCFP per se.

As mentioned earlier, the solution of MCFP will be the weights $f(v,u),\ \forall (v,u) \in E$ that represent the total number of bikes the operator has to transfer from the station that corresponds to node v to the one of node u (Fig. 2).

4 Proposed Framework

The objective of our approach is to optimize bike redistribution among the stations, in order to maximize bike availability based on the predicted users' demand. The main steps of our approach are briefly described below:

1. **Data Preparation.** We extract a time series of demand from the raw trip data. Each station is associated with a time series of bike demand values for that station for the discretization of time into intervals $\mathcal{T} = \{T_1, \ldots, T_n\}$.
2. **Time series AutoML.** We perform AutoML for each time series to efficiently search the space of forecasting algorithms and hyperparameters. This produces a predicted demand value for each station for the next time interval.

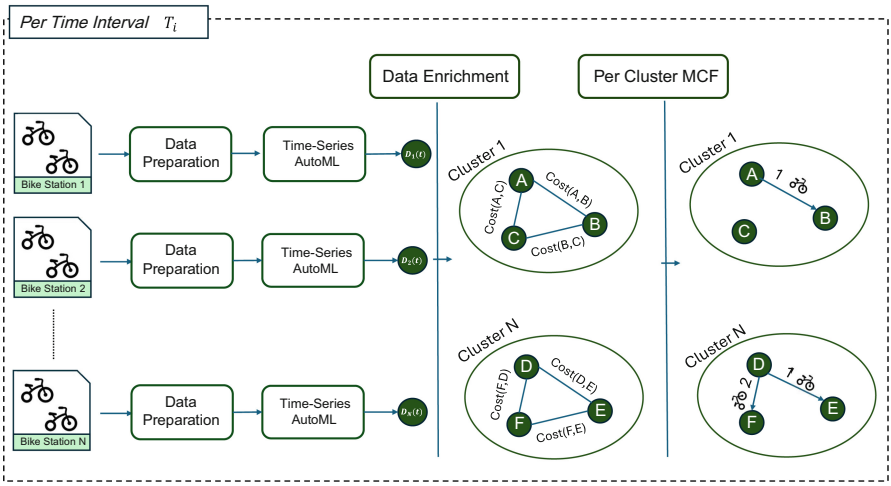

Fig. 2. The end-to-end framework for bike redistribution. For a specific time interval, we create the time series for each station and use AutoML to find the best forecasting algorithm for each station. Then, we predict the demand for each station, and for each cluster of stations, we form the graph that also captures the travel distance between stations. The solution of the MCFP determines the redistribution in each cluster.

3. **Data Enrichment.** The predicted demand values are enriched with the transportation time between stations, obtained from data of OpenStreetMap.
4. **Per Cluster MCF.** Finally, we consider k geographical clusters of stations, and build a graph for each cluster. The stations of each cluster correspond to graph nodes. The capacity is defined as the number of bikes that the operator can load to a redistribution vehicle. The cost between two stations corresponds to the estimated transportation time. Then, we solve the Minimum Cost Flow problem with relaxed constraints for every such graph.

4.1 Data Preparation

The first step is to process raw data of bike trips from one station to another to calculate the historic bike demand for each station. Each record of trips consists of (i) a trip Id, (ii) its Start Time, (iii) its End Time, (iv) the Start Station Id, and (v) the End Station Id. We partition the time into intervals $\mathcal{T} = \{T_1, \ldots, T_n\}$ and we consider the records of each group.

For every station $i \in \mathcal{S}$, we consider two time series $\mathcal{I}_i(t)$ and $\mathcal{O}_i(t)$, $t \in \{T_1, \ldots, T_n\}$, that correspond to the number of incoming and outgoing bikes during time interval t for station i, respectively. Incoming bikes are the bikes that users return to station i during an indexed time interval t, while outgoing bikes are the ones borrowed from station i during t. The time series of demand for station i for an indexed time interval t is the difference between the time series $\mathcal{O}_i(t)$ and $\mathcal{I}_i(t)$:

$$\mathcal{D}_i(t) = \mathcal{O}_i(t) - \mathcal{I}_i(t),\ t \in \{T_1, \ldots, T_n\} \tag{3}$$

Positive values of demand during t indicate that station i had a need for bikes, because the number of trips that considered i as a starting point was greater than the number of trips that considered i as an ending point. On the other hand, negative values of demand during t indicate a surplus of bikes for station i during t. Based on this, the network of stations is considered as a network that can balance itself. Stations with negative predicted demand are used in the redistribution procedure in order to provide bikes to those with positive predicted demand.

4.2 Bike Demand Forecasting

The first problem that we face is the prediction of the demand $\mathcal{D}_i(t)$ for every station i. Since the demand is represented as a time series, we need to employ time series forecasting (TSF) methods. In the literature, one can find many methods for TSF that can be broadly classified in three main categories: i) statistical methods, ii) machine learning methods and iii) deep learning methods. Each TSF method comes along with different hyperparameters, so apart from finding the best performing TSF method, we also need to find the optimal values for its hyperparameters.

AutoML for TSF (AutoTSF) seeks to solve the so-called Combined Algorithm Selection and Hyper-parameter tuning (CASH) for TSF methods. In essence, it seeks to find the TSF method and the values of its respective hyperparameters that minimizes a given error metric. Formally, let the following equation be the one to optimize:

$$\mathcal{A}^* = \arg\min_{m_\lambda} \mathcal{L}(\mathcal{D}_{train}, \mathcal{D}_{val}, m_\lambda) \tag{4}$$

where $(\mathcal{D}_{train},\ \mathcal{D}_{val}) \sim \mathcal{D}$ is a train-test split of a dataset $\mathcal{D}$, $m \in \mathcal{M}$ is a TSF method from a collection of available methods $\mathcal{M}$, $\lambda \in \Lambda_m$ is a parametric vector of the hyperparameter search space of method m, m_λ is the method m instantiated with parameters λ and $\mathcal{L}$ is a loss metric function.

In this paper, we deploy Bayesian Optimization (BO) [7] in order to solve the CASH problem. BO is a probabilistic optimization framework that optimizes black-box functions. In detail, for each time series $\mathcal{D}_i(t),\ t \in \{T_1, \ldots, T_n\}$, we select an index $r \in \{1, \ldots, n\}$ and split the time series in two parts: $\mathcal{D}_i^{train}(t),\ t \in \{T_1 \ldots, T_r\}$ and $\mathcal{D}_i^{test}(t),\ t \in \{T_{r+1} \ldots, T_n\}$. We examine two cases:

1. Consider a set of TSF methods $\mathcal{M} = \{m_1, \ldots, m_w\}$ together with the sets of their hyperparameters $\Lambda_{m_1}, \ldots, \Lambda_{m_w}$. In this case, we use the Mean Absolute Error (MAE) as loss metric to solve the optimization problem:

$$m, p = \arg\min_{m_j, p_j} MAE\left(\mathcal{D}_i^{test}, \hat{\mathcal{D}}_i^{test}|_{m_j, p_j}\right) \tag{5}$$

with BO, where $\hat{\mathcal{D}}_i^{test}|_{m_j,p_j}$ is the prediction of $\mathcal{D}_i^{test}|_{m_j,p_j}$ training the algorithm m_j with hyperparameters $p_j \in \Lambda_{m_j}$.

2. Consider a TSF method m together with the set of its hyperparameters Λ_m. In this case, we use the Mean Absolute Error (MAE) as loss metric to solve the optimization problem:

$$p = \arg\min_{p_j} MAE\left(\mathcal{D}_i^{test}, \hat{\mathcal{D}}_i^{test}|_{m,p_j}\right) \tag{6}$$

with BO, where $\hat{\mathcal{D}}_i^{test}|_{m,p_j}$ is the prediction of $\mathcal{D}_i^{test}|_{m,p_j}$ training the method m with hyperparameters $p_j \in \Lambda_m$.

4.3 Data Enrichment

Furthermore, we use the cost of transportation for any pair of stations. Specifically, we retrieve this information from OpenStreetMap [14]. For every pair of geographical points x=(longitude$_1$, latitude$_1$) and y = (longitude$_2$, latitude$_2$) that correspond to two different stations x and y, we retrieve every possible route that connects them. Afterwards, we compute for each pair of stations the shortest route, which is used to finally define $cost(x, y)$ as the cost to travel from station x to y.

4.4 Bike Redistribution

Let $\mathcal{S}$ be a set of stations and $\mathcal{D}_i(t)$, $t = \{T_1, \ldots, T_n\}$, their corresponding demand time series, as defined in Sect. 3. Let also $\hat{\mathcal{D}}_i(q)$ be the predicted demand for every station $i \in \mathcal{S}$ for the time index q that corresponds to the time interval $[t_{end}, t_{end} + \delta\mathcal{T})$, where $\delta\mathcal{T}$ is defined in Eq. 1. We consider a directed graph $\mathcal{G} = (V, E)$, where V is the set of nodes corresponding to stations and E is the set of directed vertices that indicates connections between the nodes of V.

After initializing the graph, we solve the corresponding MCFP, using as demand the values $\hat{\mathcal{D}}_i(q)$ for every node v that corresponds to a station i and by using the costs that were defined in the previous subsection. Under these assumptions, we recommend a redistribution strategy based on the predicted demand, in order to provide the operator with a plan that will cover the future demand.

The second assumption of the MCFP as defined in Sect. 3.2 is very strict, since in real-world applications (like the bike repositioning, the problem we are trying to tackle here) there are no guarantees that the total demand will sum up to zero. What we want to achieve in our case, is that outgoing flow-incoming flow will always cover the station's demand, so we can relax the second constraint as follows

$$\sum_{w \in V} f(u, w) - \sum_{w \in V} f(w, u) \geq d(u), \ \forall u \in V \tag{7}$$

So we end up with the problem's constraints defined below:

1. $f(v, u) \leq cap(v, u), \ \forall (v, u) \in E$

2. $\sum_{w \in V} f(u,w) - \sum_{w \in V} f(w,u) \geq d(u), \ \forall u \in V$
3. $f(u,v) \geq 0, \ \forall (u,v) \in E$

The replacement of the second constraint (from equality to inequality) might create artificial nodes (in the case that the total demand is greater than zero $\sum_{v \in V} d(v) > 0$), but we assume that stations have a surplus of bikes and the operator will always be able to use bikes from stations with negative demand in order to use them during the rebalancing.

Note that one could tackle the MCFP as an Integer Programming (IP) problem. Since IP problems are hard to solve, we chose to treat the MCFP as a Linear Programming (LP) problem. Generally LP problems provide continuous solutions (in our case flows of bikes to be redistributed) and for this reason, as the corresponding flows $f(v,u)$ might be continuous, we perform a rounding in order to make sure that our results are integers.

Additionally, we assume that the available stations are clustered in different subgroups. While this clustering can occur in different ways, in this paper, we utilize clustering based on the geo-location of stations. The clustering is provided by the BSS operator, but other clustering algorithms can be explored as well.

5 Experiments

In this section, we present the results of our experimentation. Our framework is tested using real-world data obtained from the Serveo Bike Sharing System operating in Barcelona for accuracy and efficiency. The framework is implemented in Python and we use the Optuna library [1] to conduct BO and the PuLP [13] library for MCFP optimization.

5.1 Experimental Setup

Dataset. Our data consist of bike trips recorded in January 2022 in the city of Barcelona. Each record consists of the starting and ending station, departure and arrival time and an associated trip id. We partition the trips in 2-h intervals and aggregate incoming and outgoing bikes per interval. There exist 469 bike stations and for each pair of stations we additionally retrieve the cost of transportation, i.e., the estimated travel time of the shortest route, as provided by OpenStreetMap.

Evaluation Methodology. We evaluate our framework in two stages. First, we explore the efficiency of the AutoML solution applied for bike demand forecasting. From the available 469 time series that correspond to each station, we select two of them that significantly differ in terms of volatility and seasonality (shown in Fig. 3), and exhaustively search their hyperparameter space through Grid Search. Additionally, we apply Bayesian Optimization (BO) over the same space, with a budget equal to that of Grid Search and compare the results.

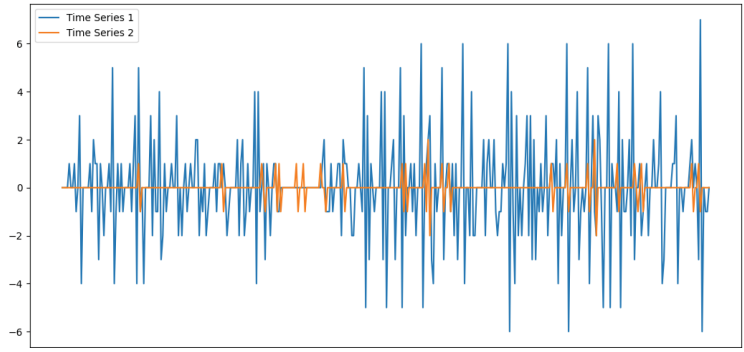

Fig. 3. Demand for two selected time series (1/1/1022–31/1/2022).

For Grid Search, continuous parametric spaces are discretized manually. On the other hand, BO may sample any point in the continuous space.

Moreover, we apply our MCF solution for each cluster of stations. For each station, we apply a narrowed-down version of AutoML based on our findings over the previous experiment. Afterwards, we forecast the demand for each station over the next 2-h time interval, by using the best individual algorithm configuration found. We then report the accuracy of the time series forecasting algorithms, visualize redistribution results and discuss our findings on how per cluster MCF may reduce operational time.

Parameters. For the AutoML in times series forecasting, we considered several types of algorithms including statistical algorithms (ARIMA, Holt Winter's Exponential Smoothing (ES), Holt Exponential Smoothing (H-ES) and Simple Exponential Smoothing (S-ES)), one deep learning algorithm (Multilayer Perceptron, MLP) and a machine learning algorithm (XGBoost Regressor, XGBoost). In Table 1, we show for each algorithm, the number of possible configurations for its hyperparameters, i.e., the size of the search space. We must also highlight, that we do not include to our experimentation other deep learning baselines, such as LSTMs, RNNs etc., due to the fact that their evaluation is empirically more computationally expensive and contradicts the operational time limits we have, even though those approaches have proven to be efficient for time series forecasting tasks.

Table 1. Number of configurations for each algorithm during Grid Search.

Algorithms	ARIMA	ES	H-ES	S-ES	MLP	XGBoost
Cardinality	1000	800	400	20	1600	15,361

Evaluation. In order to evaluate the demand forecasting algorithms, we split the time series for every station i in $\mathcal{D}_i^{train}$ and $\mathcal{D}_i^{test}$ as defined in Sect. 4.2. We

keep 80% of the original data for training and the remaining 20% for evaluation. When conducting BO, we additionally split $\mathcal{D}_i^{train}$ again accordingly to use 20% of the training data for optimization. We consider two different evaluations of accuracy on test data, $\mathcal{D}_i^{test}$:

1. n-Steps-Ahead-Forecasting (n-SAF): In this case, we evaluate the forecasting capability of the algorithm m for longer time horizons. For each $T_j,\ j > r$ we use for the prediction of the corresponding demand value, $\hat{\mathcal{D}}_i(T_j)$, the already forecasted demand values of $T_k,\ r < k < j$. Formally, this is defined as follows:

$$\hat{\mathcal{D}}_i^{test}(T_j) = m\left(\mathcal{D}_i^{train}, \hat{\mathcal{D}}_i(T_{r+1}), \ldots, \hat{\mathcal{D}}_i(T_{j-1})\right),\ \forall r < j \leq n \tag{8}$$

2. 1-Step-Ahead-Forecasting (1-SAF): In this case, we focus on short term predictions. We assume that for every $T_j,\ j > r$, the demand of $\mathcal{D}_i^{test}(T_k)$ is known $\forall r < k < j$. Formally, this is defined as follows:

$$\hat{\mathcal{D}}_i^{test}(T_j) = m\left(\mathcal{D}_i^{train}, \mathcal{D}_i(T_{r+1}), \ldots, \mathcal{D}_i(T_{j-1})\right),\ \forall r < j \leq n \tag{9}$$

Selecting an algorithm according to n-SAF methodology, is useful when we expect operational failures (e.g., stations' power outage, network's lack of communication etc.). On the other hand, algorithms selected according to their 1-SAF capability will likely be more competent but require the actual time series value at each time interval.

Metrics. We use the Mean Absolute Error (MAE), which is the absolute difference between actual and predicted demand for each station averaged over the set of time intervals. Formally it is defined as follows for each station (node) $v \in V$:

$$MAE(v) = \frac{1}{n-r-1} \sum_{t=r+1,\ldots,n} \left| d(v)_t - \widehat{d}(v)_t \right| \tag{10}$$

5.2 Experimental Evaluation of Demand Forecasting

Our findings over the extensive configuration search on the two selected time series of Fig. 3 are presented in Table 2 and Table 3 for n-SAF and 1-SAF, respectively. Our first observation is that all forecasting algorithms achieve a relatively low error.

In the case of n-SAF, we identify that the best performing algorithm is ARIMA for both time series across both strategies, with the lowest error in terms of MAE being equal to 1.5665 and 0.2133 for the two time series respectively. In the case of 1-SAF, we observe that MLP outperforms both ARIMA and XGBoost, achieving around 14.6% and 17.12% improvement over the other two, respectively. These observations imply that different algorithms may perform best depending on the application scenario.

It is also worth mentioning that some experiments conducted with BO might be slightly better than the respective Grid Search, e.g., for ES and H-ES. This is because of the discretization of the search space of continuous hyperparameters for Grid Search, whereas BO is applicable on continuous variables as such. We also note that for the 1-SAF case we used only the three algorithms that can be used for such types of predictions without the need of refitting the respective algorithms.

Table 2. Comparison of exhaustive search to BO for two time series in terms of MAE for n steps ahead forecast (n-SAF).

	Time Series 1		Time Series 2	
	Grid Search MAE	BO MAE	Grid Search MAE	BO MAE
ARIMA	**1.5665**	1.5671	**0.2133**	**0.2133**
ES	1.6027	**1.6005**	**0.2133**	**0.2133**
H-ES	1.6027	**1.6005**	**0.2133**	**0.2133**
S-ES	**1.6000**	**1.6000**	**0.2133**	**0.2133**
MLP	**1.6375**	1.6384	**0.2173**	**0.2173**
XGBoost	**1.6376**	**1.6376**	**0.2173**	**0.2173**

Table 3. Comparison of exhaustive search to BO for two time series in terms of MAE for one step ahead forecast (1-SAF).

	Time Series 1		Time Series 2	
	Grid Search MAE	BO MAE	Grid Search MAE	BO MAE
ARIMA	**1.5860**	1.5875	**0.2133**	1.0715
MLP	**1.3546**	1.5688	**0.1268**	0.1334
XGBoost	**1.6344**	1.6353	0.2167	**0.2166**

In terms of error, all algorithms showcase smaller MAE for 1-SAF than n-SAF. This result is reasonable due to the fact that in the 1-SAF case, we make the assumption that the ground truth is available, whereas on the other hand in the n-SAF case we use predicted demand. As a result, even though n-SAF may seem as a more practical predictive framework, as highlighted in the Section above, 1-SAF provides more accurate predictions, as such we adopt it in our proposed end-to-end framework.

Focusing on the on the 1-SAF case, we conducted BO only for MLP for both time series, because MLP outperforms both ARIMA and XGBoost for both time series, given a budget of 1600 trials, which is the same as for Grid Search. BO for time series 1 achieved the best MAE value of 1.3674, found in trial 386.

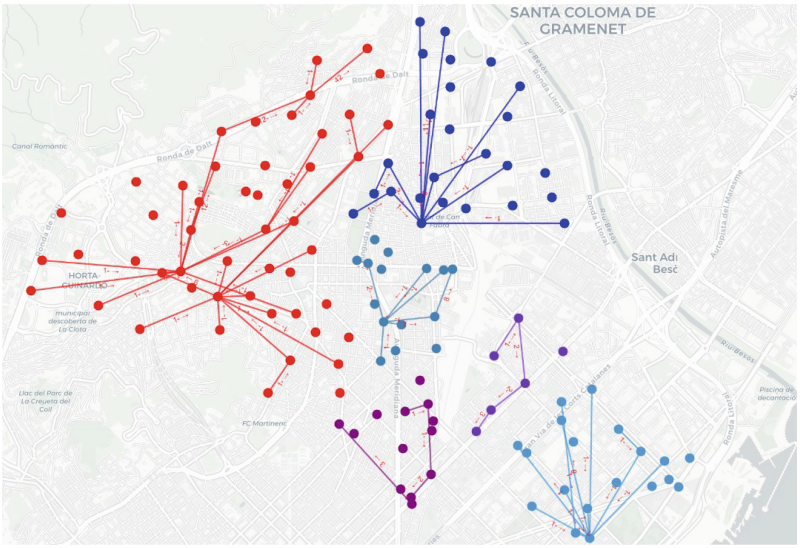

Fig. 4. A sample of redistribution per cluster of stations. Differently colored nodes represent different clusters. Edges represent the number of bikes to be transported between stations.

The respective best MAE value for time series 2 is 0.2225, found in trial 1062. Both results highlight the fact that BO is an effective solution of algorithm optimization, especially when the relative error is higher, as indicated from time series 1. For this reason, we choose to use BO for MLP for the prediction of all stations' demand future moving forward. Considering BO's execution time and in order to align with operational time limits, we chose to conduct BO for all time series using a number of trials that is equal to the 1/3 of MLP's cardinality (533 trials).

We choose to conduct BO for every station separately using MLP, as it is clear from the diverse nature of the given time series, that a single model may not be suitable for every station, so while we use a unique algorithm for all (MLP), we optimize its hyper-parameters, according to the data given.

5.3 Practical Application and Evaluation of MCF

Capitalizing on our findings over the case study on the two time series, we proceed with the evaluation of the MCF model over clusters of stations.

First, we apply for each stations' demand time series BO set to search the configuration space of MLP with a budget of 533 iterations equal to 1/3 of the trials considered in exhaustively searching MLP, based on the results discussed in the previous subsection. We then forecast the demand value for each station over the next 2-hour time interval. In Fig. 5, we showcase a box-plot of the MAE value for each station. We report the average validation MAE across every station to

be 1.0964 with standard deviation of 0.7097. Out of 464 stations, we also identify 17 that had an error rate larger than 2.5, which may be accounted to various factors such as unexpected events in the specific time interval, or general forecast difficulty due to the time series specifics. Moreover, we report that BO found the best configuration on trial number 279 on average for every station.

These predictions are afterwards provided along with the estimated duration for each trip among a pair of nodes to our MCF based approach. We identify the station clusters according to their geographic locations, and apply MCF for each of these clusters. The resulting bikes to be transported are then retrieved, as shown in Fig. 4 just for a sample of the available clusters.

Finally, we compare the execution time of the MCF solving algorithm when applied to each cluster against running MCF over all stations (i.e., 1 cluster). We report the average execution time of per cluster MCF to be around 0.38 s. The total execution time for per cluster MCF is 15.64 s, against 8.54 min of execution time when applying MCF over all stations. We note the magnitude of difference between the two approaches, especially since long execution times may be prohibitive when there is need for redistribution in short time intervals.

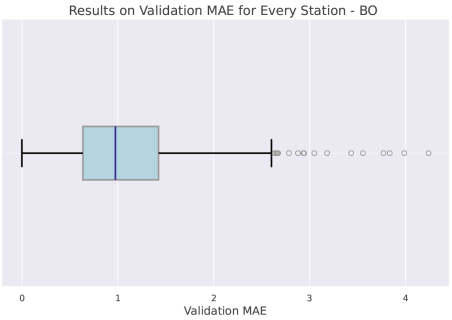

Fig. 5. Best validation MAE achieved for every station after Bayesian Optimization

6 Conclusions and Future Work

In this paper, we proposed an AutoML-based approach for bike demand forecasting and redistribution. To address the first problem of bike demand forecasting, we adopt an AutoML approach that uses Bayesian Optimization in order to find the best time series forecasting method and values of its hyperparameters. Then, the second problem is to determine the process of bike redistribution among stations to cover the predicted demand. To this end, we cast the problem of bike redistribution to the Minimum-Cost Flow Problem to find the optimal solution. Our experiments with real-world data show that careful use of AutoML can indeed find a well performing configuration close to the optimal much faster

than traditional approaches, and that a meaningful redistribution can be computed. In our future work, it would be interesting to deploy our redistribution in operational settings to evaluate the improvement of the bike sharing service and user satisfaction. Additionally, future work may include modeling of the demand that is unknown to the system due to the unavailability of bikes in a station, taking also into consideration other validation metrics to align with the robustness of the data.

Acknowledgements. This work has received funding from the Horizon Europe research and innovation programme under the GA 101070416 (project Green.DAT.AI).

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of SIGKDD, pp. 2623–2631, 2019
2. Babicheva, T., Burghout, W.: Empty vehicle redistribution in autonomous taxi services. EURO J. Transp. Logist. **8**(5), 745–767 (2019). https://doi.org/10.1007/s13676-019-00146-5
3. Dell'Amico, M., Hadjicostantinou, E., Iori, M., Novellani, S.: The bike sharing rebalancing problem: mathematical formulations and benchmark instances. Omega **45**, 7–19 (2014)
4. Deng, D., Karl, F., Hutter, F., Bischl, B., Lindauer, M.: Efficient automated deep learning for time series forecasting. In: Proceedings of PKDD, pp. 664–680, 2022
5. Fishman, E.: Bikeshare: a review of recent literature. Transp. Rev. **36**, 92–113 (2016)
6. Gan, J., Zhang, G., Zhang, Y.: Bike rebalancing: how to find a balanced matching in the k center problem? Eur. J. Oper. Res. **316**, 845–855 (2024)
7. Garnett, R.: Bayesian Optimization. Cambridge University Press, Cambridge (2023)
8. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning - Methods, Systems, Challenges. Springer (2019)
9. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. J. Stat. Softw. **27**(3), 1–22 (2008)
10. Karamanis, R., Anastasiadis, E., Stettler, M., Angeloudis, P.: Vehicle redistribution in ride-sourcing markets using convex minimum cost flows. IEEE Trans. Intell. Transp. Syst. **23**, 10287–10298 (2022)
11. Liang, J., Jena, S. D., Lodi, A.: Dynamic rebalancing optimization for bike-sharing systems: a modeling framework and empirical comparison. Eur. J. Oper. Res. **317**, 875–889 (2024)
12. Liu, J., Sun, L., Chen, W., Xiong, H.: Rebalancing bike sharing systems: a multi-source data smart optimization. In: Proceedings of SIGKDD, pp. 1005–1014, 2016
13. Liu, J., Sun, L., Chen, W., Xiong, H.: PuLP: a linear programming toolkit for Python (2011)
14. OpenStreetMap contributors. Planet dump retrieved from, 2017. https://planet.osm.org, https://www.openstreetmap.org

15. Regue, R., Recker, W.: Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. Transp. Res. Part E: Logist. Transp. Rev. **72**, 192–209 (2014)
16. Shchur, O., et al.: Autogluon-timeseries: automl for probabilistic time series forecasting. In: Proceedings of AutoML, pp. 9/1–21, 2023
17. Wang, Y.-J., Kuo, Y.-H., Huang, G.Q., Weihua, G., Yaohua, H.: Dynamic demand-driven bike station clustering. Transp. Res. Part E: Logist. Transp. Rev. **160**, 102656 (2022)
18. Zimmer, L., Lindauer, M., Hutter, F.: Auto-pytorch: multi-fidelity metalearning for efficient and robust autodl. IEEE Trans. Pattern Anal. Mach. Intell. **43**, 3079–3090 (2021)