Transportation Research Part C 128 (2021) 103213

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Deep survival modelling for shared mobility

Bojan Kostic, Mathilde Pryds Loft, Filipe Rodrigues *, Stanislav S. Borysov

Machine Learning for Smart Mobility group[1], *Transport Division, DTU Management, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark*

ARTICLE INFO

Keywords:
Shared mobility
Car-sharing
Data Science
Survival analysis
Neural networks
Deep survival modelling

ABSTRACT

With an increased focus on minimising traffic externalities in metropolitan areas, a growing interest in environmentally friendly mobility systems has emerged, such as electric car-sharing systems. However, increasing demand and larger service areas often make it difficult to keep cars available where and when customers need them. This problem can be alleviated by predicting for how long cars stay vacant at given pick-up/drop-off locations. To maximise their usage, shared fleet operators relocate the cars to more desired locations. In this paper, we tackle the problem of predicting time to pick-up for shared cars in a probabilistic way by applying time-to-event modelling through survival analysis. Both statistical and machine learning approaches to survival regression are investigated. In addition, we propose the use of Gaussian copulas in order to model the correlation among vacant vehicles and to obtain more refined event-based predictions. First, an exploratory analysis is done to investigate the effect of various features on car vacancy time, which can provide significant insights into vacancy times and their influencing factors. Second, the Cox proportional hazards model (CPH), a linear survival model, is compared to DeepSurv, a neural-network-based survival model. To predict survival times, a two-step approach is formulated: in the upper level, a classification model is used to classify cars based on vacancy time duration and, in the lower level, time-to-event modelling is applied to each class using independent survival analysis models. Our empirical results using data from Copenhagen demonstrate that the DeepSurv model leads to a stronger fit compared to CPH. Moreover, we were able to verify that the proposed two-step approach can result in an improvement of over 15 % in performance compared to a standard one-step approach. Lastly, we demonstrate the benefits of survival models for relocation optimisation.

1. Introduction

Shared mobility services, and especially car-sharing services, have been increasingly introduced in cities in the recent years. Shared mobility is a rapidly increasing form of transportation worldwide, with a goal to reduce traffic congestion and provide more environmentally friendly transportation systems. There are various types of shared mobility concepts, such as car-sharing (station-based and free-floating), bike-sharing or on-demand services. For a broad overview of shared mobility, see, for example,Soares Machado et al. (2018).

Shared mobility systems have attracted growing attention of the research community, primarily driven by increasing numbers of private and public companies entering the market. A wide range of approaches has been applied to study the car-sharing market, starting from discrete choice modelling for individuals to join car-sharing services (Juschten et al., 2019) to modelling and simulation

* Correspondence to: Bygningstorvet 116B, 2800 Kgs. Lyngby, Denmark.
E-mail addresses: boko@dtu.dk (B. Kostic), maplo@adm.dtu.dk (M.P. Loft), rodr@dtu.dk (F. Rodrigues), stabo@dtu.dk (S.S. Borysov).
[1] MLSM

https://doi.org/10.1016/j.trc.2021.103213
Received 27 July 2020; Received in revised form 4 May 2021; Accepted 5 May 2021
Available online 11 May 2021

of car-sharing competition (Balac et al., 2019). Another important aspect is the ongoing research on the system-wide impact of the shared mobility. Shared mobility systems have been analysed from both simulation and data-driven perspectives as more data become available from private fleet operator companies. Numerous studies have shown the positive effect of introducing shared mobility concepts in the existing transport system. For instance, Becker et al. (2020) conducted a joint simulation of car-sharing, bike-sharing and ride-hailing systems for Zurich, Switzerland, to assess the welfare impact of these services. They shown that these services can increase the overall system's efficiency in terms of travel times and costs, while at the same time substantially reducing energy consumption. At the same time, the overall impact of such systems remains unclear (Erhardt et al., 2019).

However, with increasing demand and larger service areas, shared mobility companies start facing challenges related to keeping the vehicles available to customers at their desired locations. This issue is amplified when the fleet operates in a free-floating regime, i.e. when the customers are free to drop off the cars where it is most convenient for them. One solution to tackle this issue is to relocate cars manually to the locations with a higher demand. Being able to provide reliable predictions of the vacancy times of cars can be used for optimisation of the shared vehicles fleet usage and lead to higher user satisfaction.

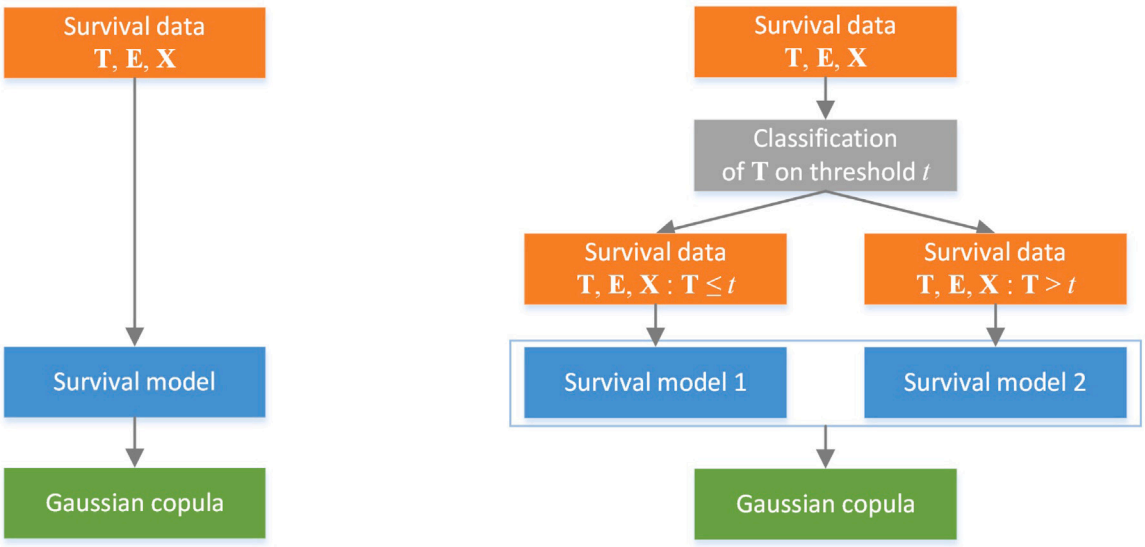
Vosooghi et al. (2017) provide an overview of travel demand estimation approaches for car-sharing systems, most of which are based on activity-based multi-agent simulation and time series analysis. Schmöller et al. (2015) provide an empirical study for Munich, Germany and Berlin, Germany to identify influencing factors for car-sharing demand. Müller and Bogenberger (2015) preform a time series analysis of booking data using an ARIMA model for the case study of Berlin. Sohi (2018) applied an ARIMA model and neural networks for time series analysis for demand forecasting, i.e. predicting the expected number of bookings of car-sharing services, with a case study in Munich, Germany, using DriveNow car-sharing company data. Wang et al. (2019) investigate the optimisation of dynamic relocation operations for one-way car-sharing fleets consisting of electric vehicles. On the modelling side, most of the research about car-sharing is focused on demand prediction and vehicle relocation; e.g. Ciari et al. (2014) model both station-based and free-floating car-sharing demand in Berlin, Germany. As free-floating car-sharing systems allow vehicles to park anywhere in a designated area, Balac et al. (2017) investigate the effect of parking pricing on the case of Zurich, Switzerland.

Research on bike-sharing has attracted more attention than car-sharing, possibly because more open bike-sharing datasets are freely available. Here the focus is again on bike demand and supply prediction and rebalancing. In terms of the methods, recent demand prediction approaches for bike-sharing systems are mostly based on supervised machine learning methods (Pereira and Borysov, 2018; Qian et al., 2018; Ashqar et al., 2017). Some of them use dynamic linear models (Almannaa et al., 2020), random forests (Ruffieux et al., 2017) and, more commonly, various types of deep neural networks architectures, such as recurrent neural networks (which are typically used for sequence data and time series analysis), with two most common types being long short-term memory (LSTM) and gated recurrent units (GRU) (Pan et al., 2019; Wang and Kim, 2018), convolutional neural networks (Ruffieux et al., 2017), Gaussian processes (Li and Zheng, 2019) and recently censored Gaussian processes (Gammelli et al., 2020).

The methodology proposed in this paper is general and can be applied to any type of shared mobility system besides car-sharing, such as bike-sharing or e-scooter-sharing. Most of the previously mentioned approaches are mainly focused on time-varying demand that is aggregated both spatially and temporally (e.g. number of bookings per 15 min per zone), which contains errors caused by the spatial and temporal discretisation. In contrast, to avoid this, our approach is disaggregated, treating vehicles individually and predicting single-vehicle availability (from a drop-off to the next pick-up). The core idea of the analysis is to represent the car vacancy time as *survival time*. Unlike other machine learning methods, survival analysis is particularly designed to handle censored observations, which are intrinsically part of car sharing, due to vehicle relocations. Survival analysis, and especially deep survival analysis, has gained traction recently due to fast developments in artificial intelligence (Lee et al., 2018). Beside the medical applications, where it was applied originally, it is now applied in the context of life-course calendars, which are used to analyse mobility biographies (Kostic et al., 2020), for predicting credit risk (Bellotti and Crook, 2009), customer churn (Tang et al., 2007), predictive maintenance (Finkelstein, 2008) and many other applications (see Lee and Whitmore, 2006).

The main goal of this paper is to explore new methods for predicting vacancy time of shared cars. It aims to investigate how well the methods from survival analysis and Gaussian copulas can be applied in this context and, subsequently, to provide comparison among different models. Several survival models, namely non-parametric, linear and neural-network-based survival models, will be applied here in the shared mobility context for the first time. Two approaches will be tested: a standard one-step approach and a two-step approach. The former is a simple approach where survival analysis is directly applied to the whole dataset, while the latter introduces a classification layer to divide the data into two groups, for which two separate survival models will be trained. Several machine leaning models for classification are tested. For both approaches, once survival predictions are obtained, we apply Gaussian copulas to embed spatial correlations among the vehicles and quantify its effect. Our empirical results show that the two-step approach shows a great increase in performance comparing to the one-step approach. The main contributions of this paper are the application of survival models (using censored observations) to estimate the distribution of vehicles' vacancy times, and the use of Gaussian copulas to spatially correlate individual vacancy time distributions for nearby vehicles. In addition, the exploratory study shows interesting results about the factors influencing the pick-up times. The results show that survival analysis can be a valuable tool in analysing shared mobility data.

The remainder of the paper is organised as follows. Section 2 explains the mathematical methods used in the paper, specifically survival analysis in modelling time-to-event data, machine learning models for classification, and Gaussian copulas. Section 3 describes the dataset, data cleaning and pre-processing procedures. Section 4 provides a high-level exploratory analysis of the factors influencing survival times using simple survival models. Section 5 presents the results and discussion on the application of survival regression models and Gaussian copulas for vacancy time prediction. Section 6 benchmarks the proposed survival analysis approach against popular regression models from the literature. Finally, Section 7 contains conclusions and outlines directions for future work.



(a) One-step approach. (b) Two-step approach.

Fig. 1. Modelling approaches. (a) The one-step approach directly models survival time distribution. (b) The two-step approach fits two survival models based on the car vacancy duration given by the classifier.

2. Methodology

We present here the general overview of the methodology, followed by the description of each individual part of it.

2.1. Modelling approaches

Let us consider a dataset that consists of a set of features **X** (which are also called covariates, explanatory variables, independent variables or predictors) and labels **y** (also called dependent variables or response variables). The matrix **X** is of size $n \times m$, n being the number of observations (rows) and m being the number of features (columns). The **y** in our case consists of two vectors—survival times **T** and censorship event indicator **E**. Survival time represents car vacancy time (drop-off to pick-up). Censorship indicator describes whether the survival time is fully observed (uncensored) or partially observed (censored due to vehicle relocation). For this dataset, two approaches will be tested: the one-step approach and the two-step approach. A schematic presentation of the two approaches is shown in Fig. 1.

The one-step approach, depicted in Fig. 1a, can be considered a typical prediction problem, where we take a prediction model and apply it to the entire dataset. The difference with respect to classical regression problems, where the prediction is a point estimate of the response variable, is that here we estimate the distribution of the response variable. The prediction models here are different survival models (Section 2.2): the models take as input the feature matrix **X** and predict the probability distribution over the response variable **T**, taking into consideration the censorship indicator **E**.

In the two-step approach, depicted in Fig. 1b, we first fit a classifier (Section 2.3) which predicts whether the vacancy time is going to be longer that a certain threshold value, for example, 12, 24 or 48 h. This gives us two separate dataset. Then, we apply separate survival models (Section 2.2) to each one of them, as in the one-step approach. The predictions from the two survival models are evaluated using the concordance index (C-index) described in Section 2.2.4.

Lastly, for both approaches, once we obtain predicted individual survival functions, they are then embedded into Gaussian copulas (Section 2.4) to create the spatial correlation among close vehicles. The output of the copulas are adjusted survival functions, as a consequence of a vehicle being picked up or dropped of in their vicinity.

To prove the applicability of the achieved predictions for deploying policy or operational measures, we test a simple minimum cost flow problem (Appendix) for vehicle relocation optimisation.

2.2. Survival analysis

Survival analysis (Harrell, 2015) provides time-to-event (TTE) analysis with censored observations. TTE is expressed through the two events: the *birth* event (which in our case is the car drop-off event) and the *death* event (which is the subsequent car pick-up event), hence the term “survival”. The time between the two events is the survival time, denoted by T, which is treated as a non-negative random variable. Given the birth event, the survival function represents the probability distribution of the death event

happening in time. In addition, observations of true survival times may not always be available, thus creating distinction between uncensored observations (true survival times are known) and censored observations (true survival times are unknown, but only the censorship time). This information is expressed through event indicator E with binary outcome. The censorship time is assumed to be non-informative, and T and E are independent.

The distribution of T can be described by the probability density function (PDF), denoted by $f(t)$, the cumulative distribution function (CDF), denoted by $F(t)$, and the survival function $S(t)$. The survival function can be defined from the cumulative distribution function as $S(t) = 1 - F(t)$. They can be formulated in terms of probabilities: $F(t)$ is the probability of the event happening before time t, as in Eq. (1), and $S(t)$ is the probability of the event happening after time t, as in Eq. (2), i.e. the probability of surviving time t.

$$F(t) = \mathrm{P}(T \leq t), \tag{1}$$

$$S(t) = \mathrm{P}(T > t). \tag{2}$$

Since in survival analysis we do not get exact survival time predictions but the time-dependent probabilities, those are computed through the time-dependent hazard rates. The hazard rate $h(t)$ is defined as the risk of dying in an infinitesimally small period of time. It can be formulated as the probability of the event occurring in the period from t to $t + \Delta t$, given that the subject already survived up until t, scaled by the size of Δt, as shown in Eq. (3). As Δt goes towards zero $h(t)$ becomes the slope of the tangent, describing the instantaneous risk of dying at exactly time t. The survival function is then computed using Eq. (4), with H being the cumulative hazard function.

$$h(t) = \lim_{\Delta t \to 0} \frac{\mathrm{P}(t < T \leq t + \Delta t \mid T > t)}{\Delta t}, \tag{3}$$

$$S(t) = \exp\left(-\int_0^t h(t)dt\right) = \exp\left(-H(t)\right). \tag{4}$$

This is important as we can now interpret survival functions in terms of the gradient (slope) of the curve, with steeper line segments meaning an event is more likely to happen in the corresponding interval.

Univariate survival models and survival regression are typically used in survival analysis. The models we use in this paper are described below.

2.2.1. Kaplan–Meier estimator

The Kaplan–Meier estimator (Kaplan and Meier, 1958) is a univariate survival model that takes as input only observed survival times with event indicator, i.e. $S(t) = f(\mathbf{T}, \mathbf{E})$, and produces a survival function that describes the average behaviour of the whole population. It is a non-parametric counting-based method shown in Eq. (5):

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{d_i}{n_i}\right), \tag{5}$$

where d_i is the number of death events at time t_i and n_i is the total number of observations for which at time t_i the event still did not happen, i.e. that are at risk of happening. Applied to our case, d_i is the number of pick-ups at time t_i, and n_i is the total number of cars still not picked up by time t_i.

Univariate survival models are useful to get population-level insights into survival behaviour, but cannot be used for observation-level analysis as they do not take into account the independent variables, also known as covariates or features, for each observation. To consider the effect the features have on survival outcome, survival regression models are used.

2.2.2. Cox proportional hazards model

The Cox proportional hazards model (CPH) (Cox, 1972) is the most widely used survival regression model. It is a semi-parametric model that estimates the log-hazard rate (i.e. the risk) as a linear combination of the features as $\boldsymbol{\beta}^\top \mathbf{x}_i$, where $\boldsymbol{\beta}$ is a vector of coefficients and $\mathbf{x}$ is a vector of feature values for the observation i. The proportional hazards assumption means that the ratio of hazard rates of two observations is always constant, as can be seen from Eq. (6), and depends only on the time-constant feature values:

$$h(t|\mathbf{x}_i) = h_0(t) \exp\left(\boldsymbol{\beta}^\top \mathbf{x}_i\right), \tag{6}$$

where $h_0(t)$ is the baseline hazard, which vary over time and it is the same for all observations. Therefore, an estimate for the probability of surviving longer can be obtained from the partial hazard alone, as this differs for the individual observations. The higher the partial hazard the lower the survival probability. The baseline hazard is usually modelled using the Breslow estimator (Lin, 2007), the non-parametric maximum likelihood estimator for the cumulative baseline hazard function. Therefore, the linear combination will yield the estimated risk value, which is then used to compute time-dependent hazard rates, which are used to compute survival function, using Eq. (4).

CPH is trained using the Cox partial likelihood. The partial likelihood equation is shown in Eq. (7). It is the product of the specific hazard ratios for all uncensored observations. The hazard ratio for an observation i with survival time T_i represents the ratio of its

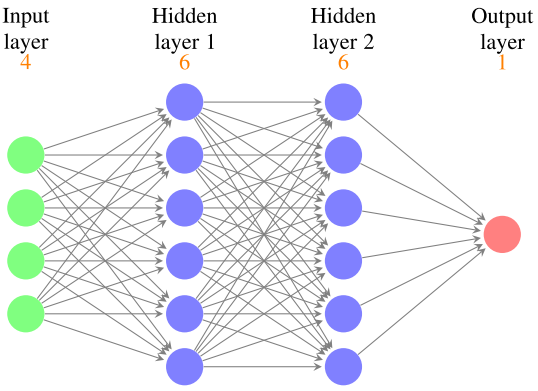


Fig. 2. A schematic structure of the multi-layer perceptron (MLP) neural network used in DeepSurv.

partial hazard to the sum of partial hazards for the observations where the event happened at the same time or after its event time, i.e. the set of observations survived by time T_i,

$$L(\boldsymbol{\beta}) = \prod_{i:E_i=1} \frac{\exp\left(\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}_i\right)}{\sum_{j:T_j \geq T_i} \exp\left(\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}_j\right)}. \tag{7}$$

2.2.3. DeepSurv

DeepSurv model (Katzman et al., 2018) is a survival model based on a deep feed-forward artificial neural network. Its neural network architecture for risk estimation is a multi-layer perceptron (MLP), consisting of an input layer, a number of hidden layers and an output layer. The input layer takes (standardised) features, the hidden layers apply non-linear transformations and the output layer consists of a single node, producing a predicted risk value for the corresponding observation. A schematic general structure of a multi-layer perceptron network is shown in Fig. 2, with four input nodes, two hidden layers with six nodes each and a single output node. Note that later in the experiments, there will be many different variations of MLPs trained for each one of the investigated scenarios, where each one of them potentially has a different number of input nodes and hidden layers and nodes, which is based on the hyper-parameter tuning.

DeepSurv, as CPH, is a proportional hazards model, which maximises the likelihood function that is generally the same as for the CPH model, with the difference that the linear relation $\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}_i$ is replaced with a nonlinear relation $f(\mathbf{W}, \mathbf{x}_i)$, where $f(\cdot)$ is the neural network, as in Eq. (8). The estimated risk value from the output node is used to compute the hazard function, and consequently the survival function, as with CPH.

$$L(\mathbf{W}) = \prod_{i:E_i=1} \frac{\exp\left(f(\mathbf{W}, \mathbf{x}_i)\right)}{\sum_{j:T_j \geq T_i} \exp\left(f(\mathbf{W}, \mathbf{x}_j)\right)} \tag{8}$$

Neural network training is done using an optimisation algorithm, usually a gradient decent method. A prediction is obtained by propagating input values forward through the network. The true and predicted values are compared and the error is calculated. The update of the model weights is done using the back-propagation algorithm (Rumelhart et al., 1986b). It propagates the gradient backward through all the hidden layers and nodes using the chain rule. It is therefore an iterative procedure that terminates when the optimisation converges.

Deep neural network applications require the tuning of a number of hyper-parameters. These parameters affect how the network is built and how it learns. Typical hyper-parameters related to the network structure include the number of hidden layers, the number of nodes per hidden layer, activation function for each layer, bias and weights regularisation, dropout and batch normalisation. Common hyper-parameters related to training include the choice of the optimisation algorithm (with its parameters, usually the learning rate), the batch size and the number of epochs (i.e. iterations, runs over the complete training dataset). The hyper-parameters are usually tuned with a grid search approach, hence this procedure can be very computationally intensive.

2.2.4. Performance metrics

The concordance index (C-index) (Harrell, 2015) is commonly used as the main metric of a survival model's quality and performance. It is a rank correlation score that measures the correctness of the ordering of predicted survival times (or hazard rates) against the observed times. For the predicted survival time the median or mean survival time is typically used. A value of 1 represents a perfect score, random guessing corresponds to 0.5 (Harrell, 2015). Eq. (9) shows the C-index calculation, where $f(x)$ is a model's predicted survival time or hazard rate,

$$C = \frac{1}{n} \sum_{i:E_i=1} \sum_{j:T_j \geq T_i} \mathbf{1}_{f(x_i) < f(x_j)} \tag{9}$$

Here, the indicator function counts the times a model predicts $f(x_i) < f(x_j)$ when observed $T_i < T_j$ holds true. The total number of comparable cases is denoted by n.

2.3. Classification

Classification will be used to classify data based on specified threshold imposed on observed vacancy times. It is a binary classification task with two outcomes: class 1 — vacancy times are less than or equal to the specified threshold and class 0 — vacancy times are greater than the specified threshold. Three machine learning models are compared: logistic regression, multi-layer perceptron and gradient-boosted decision trees. The goodness-of-fit metric used is the F_1 score, which represents a harmonic mean between the model's precision and recall as follows:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}. \tag{10}$$

Here, TP is the number of true positives (correctly classified samples of the first class), TN is true negatives (correctly classified samples of the second class), FP is false positives (samples wrongly classified as the first class) and FN is false negatives (samples wrongly classified as the second class). Since all the models and performance metric are well-established, we provide here only a brief description and recommend readers to follow the references for detailed explanations.

2.3.1. Logistic regression

Logistic regression (LR) (Hastie et al., 2017) is a linear model used for classification. It is similar to linear regression, whose output is passed through a logistic function (or sigmoid function), thus mapping real values to a range between 0 and 1. The class is assigned according to the decision boundary, which is usually set to 0.5. It is a white-box model as the trained coefficients (beta values) can then be directly interpreted, the same way as for a standard linear regression model. The model is usually trained using a gradient decent optimisation algorithm by maximising the likelihood function based on binary cross-entropy. Hyper-parameters for tuning usually include regularisation type, strength of regularisation and classes weight in the case of unbalanced data.

2.3.2. Multi-layer perceptron

Multi-layer perceptron (MLP) (Rumelhart et al., 1986a) is a standard deep feed-forward neural network that can be used for regression and classification tasks. As explained in the DeepSurv section, there are many hyper-parameters to be tuned. The neural network will use a logistic activation function for the output layer. It optimises the cross-entropy loss function by the use of back-propagation and uses the softmax function to calculate the probabilities in the output layer. This neural network without any hidden layers becomes equivalent to the logistic regression.

2.3.3. Gradient-boosted decision trees

The gradient-boosted decision trees (GBDT) (Friedman, 1999) classifier is an ensemble estimator that fits a number of single estimators, in this case decision trees, on various sub-samples of the dataset. A decision tree is a non-parametric method that works by creating simple decisions rules on which the predictions are based. It is easy to interpret as decision trees can be visualised. It is the main building block of various ensemble methods. The GBRT builds the estimators sequentially, where each following tree is fitted to the residuals from the previous trees. Typical hyper-parameters for tuning include the number of estimators and the maximum depth of the tree.

2.4. Gaussian copulas

To correlate neighbouring vehicles, such that a change in one vehicle's status (picked up or dropped off) affects the survival functions of the other vehicles, we apply Gaussian copulas (Sklar, 1973). To define a copula, let us start by defining a multivariate Gaussian distribution $\mathbf{X}$ of dimensionality k (i.e. it consists of k random variables), with the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, such that $\mathbf{X} \sim \mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The distribution is centred around zero, i.e. $\boldsymbol{\mu} = \mathbb{E}[\mathbf{X}] = \mathbf{0}$. All marginal probability density functions (PDFs) $\phi(x_i)$ follow a standard normal distribution, i.e. $x_i \sim \mathcal{N}(0,1)$, $i = 1, \ldots, k$. Their respective cumulative distribution functions (CDFs) are denoted by $F_{Gi}(x_i)$, and are uniformly distributed, such that $u_i(F_{Gi}(x_i)) \sim U(0,1)$. These are the building blocks that construct a Gaussian copula C_G, which we can define as a joint multivariate Gaussian distribution with uniform marginals:

$$C_G(u_1, \ldots, u_k) = \Phi(u_1, \ldots, u_k; \boldsymbol{\Sigma}) \tag{11}$$

with Φ being the joint CDF. We can now use the uniform marginals as a connection point to create a map between the distributions. On the copula side, the uniform marginals are obtained from the Gaussian CDF. On the other side, we can use the same uniform marginals to map them to CDFs of arbitrary distributions. The arbitrary distributions in our case are estimated survival distributions with survival functions S_i, where $S_i = 1 - F_i$, F_i being the CDF. In this way we have a connection between our survival distributions S_i and univariate Gaussian distributions $\phi(x_i)$, which in fact construct the copula. The covariance matrix of the Gaussian copula is where we embed the correlations among the random variables. Therefore, we use copulas to create a joint distribution between vehicles' survival functions that are spatially correlated.

In our approach, we start with estimating independently all the marginal survival distributions (S_i) using deep survival models explained in Section 2.2.3. Here we use an engineered feature representing the number of neighbouring vehicles in a defined radius around the vacant car, to approximately account for influencing vehicles effect. We can then derive marginal CDFs (F_i), which give us uniformly distributed data (u_i). This is directly connected to copulas marginals $F_{Gi}(x_i)$, which are also uniformly distributed.

Copula is then the density of the transformed random variables. The procedure is presented below, where steps −2, −1 and 0 are part of deep survival modelling and steps 3, 2, 1 and 0 are part of Gaussian copula.

$$\begin{array}{ccccccc|c} S_1 & \longleftrightarrow & F_1 & \longleftrightarrow & u_1 & \longleftrightarrow & F_{G1} \longleftrightarrow \phi_1 \longleftrightarrow & \\ S_2 & \longleftrightarrow & F_2 & \longleftrightarrow & u_2 & \longleftrightarrow & F_{G2} \longleftrightarrow \phi_2 \longleftrightarrow & \boldsymbol{\Phi}(\mathbf{0}, \boldsymbol{\Sigma}) \\ & & & & \vdots & & & \\ S_k & \longleftrightarrow & F_k & \longleftrightarrow & u_k & \longleftrightarrow & F_{Gk} \longleftrightarrow \phi_k \longleftrightarrow & \\ -2 & & -1 & & 0 & & 1 \quad 2 & 3 \end{array}$$

In the probabilistic sense, as survival function and CDF are cumulative quantities, sometimes is more intuitive to work with absolute quantities, such as PDF, hence the survival part of the upper procedure can be easily extended as the following:

$$\begin{array}{ccccc} S_i & \longleftrightarrow & F_i & \longleftrightarrow & u_i \\ & & \updownarrow & & \\ & & f_i & & \end{array}$$

One of the important characteristics of a multivariate Gaussian distribution is that, while the joint distribution is a Gaussian, its every conditional distribution is also a Gaussian (which is not the case with other continuous distributions), which is a property that is exploited here. By having an event on one random variable (i.e. when one out of several neighbouring cars is picked up or a new car is dropped off), we condition on that time instant and update the other random variables (i.e. through the copula we update the survival distributions of the remaining cars). We therefore dynamically adjust the survival distributions of the cars. We thus obtain an event-based model that dynamically adjusts itself. In this way, we overcome the limitation of time-constant features used in survival function estimation and obtain a more precise survival time predictions that are dependent on the other cars. An example of the procedure is outlined below, where as the vehicle 2 is being picked up, the change reflected in its survival function is propagated to the copula, from where the other survival functions are updated.

$$\begin{array}{ccccccc|c} S_1 & \longleftarrow & F_1 & \longleftarrow & u_1 & \longleftarrow & F_{G1} \longleftarrow \phi_1 \longleftarrow & \\ S_2 & \longrightarrow & F_2 & \longrightarrow & u_2 & \longrightarrow & F_{G2} \longrightarrow \phi_2 \longrightarrow & \boldsymbol{\Phi}(\mathbf{0}, \boldsymbol{\Sigma}) \\ & & & & \vdots & & & \\ S_k & \longleftarrow & F_k & \longleftarrow & u_k & \longleftarrow & F_{Gk} \longleftarrow \phi_k \longleftarrow & \end{array}$$

The update through the copula is based on the Sklar's theorem (Sklar, 1973), which states that any joint distribution can be written in terms of its copula and its univariate marginals. In the simplest bivariate case, the joint probability of x_1 and x_2 can be written as $p(x_1, x_2) = p_1(x_1)p_2(x_2)c(u_1, u_2)$, where p_1 and p_2 are marginal distributions and c is a copula with $u_1 = F_i(x_i)$. We know that if two random variables are independent it holds that $p(x_1, x_2) = p_1(x_1)p_2(x_2)$, which is a special case of the copula equation with $c(u_1, u_2) = 1$, i.e. two variables are independent if they have a constant copula of 1. To generalise our approach to all the vacant vehicles in the system, we can assume that the vehicles outside the defined radius have a copula of 1, without explicitly computing it.

Regarding the correlation structure, ideally it would be embedded into the estimation procedure of the marginal survival functions, to obtain intrinsically correlated multi-output survival model and to account for more precise time-dependent spatial correlation effects. This would, however, require designing a new model architecture capable of handling varying number of vehicles or creating a multi-output model outputting distributions for all vehicles separately. This would be a very large and computationally intensive model, hence in this research we focus on the presented approach above with using copulas sequentially.

3. Data

The data used in this research was kindly provided by SHARE NOW Copenhagen (Share Now, 2020) (called "DriveNow" at the time of collecting the data DriveNow, 2019), a car-sharing company. The data comes from a fleet of electric cars operating as a free-floating car-sharing system.

The dataset consists of several hundred thousand observations, where each observation is one realised trip. A trip is described by a set of features, such as vehicle ID, pick-up and drop-off data (location, time and battery level). The dataset includes hundreds of cars in the time period of around a year, collected in 2017 and 2018. An example of an observation from the dataset is shown in Table 1.

The allowed area for car locations covers most of the city, and is divided into 98 zones of various granularity, with the average size of 1.18 sq. km. The zone ID can be inferred from the pick-up/drop-off location coordinates. The survival model which uses the zone ID as the only feature will be used as a baseline.

In order to apply survival analysis, survival times for the cars need to be calculated. As described in Section 2.2, survival time is the time between the two events: the "birth" event, which in our case is the drop-off event, and the "death" event, which is the subsequent pick-up event. By simply subtracting the drop-off time from the subsequent pick-up time we obtain the survival time. Therefore, survival time can be defined as the time a car is vacant, i.e. the idle time of the car. Fig. 3 shows the distribution of the observed survival times for 15-minute periods. The maximum survival time encountered in the dataset is over 200 h, hence the x-axis of the plot has been adjusted for readability to 50 h.

Table 1
An example of an observation from the dataset.

Field	Value	Unit
Vehicle ID	25	ID
Time — booking	2017-08-02 12:40:00	Timestamp (yyyy-mm-dd HH:MM:ss)
Time — pick-up	2017-08-02 12:53:00	Timestamp (yyyy-mm-dd HH:MM:ss)
Time — drop-off	2017-08-02 13:16:00	Timestamp (yyyy-mm-dd HH:MM:ss)
GPS location — pick-up	(55.688111, 12.579788)	(Latitude, Longitude)
GPS location — drop-off	(55.786024, 12.521814)	(Latitude, Longitude)
Battery level — pick-up	68	%
Battery level — drop-off	32	%
Zone ID — pick-up	1	ID
Zone ID — drop-off	2	ID

Table 2
The distribution of the data based on survival times.

$T>$	Observations	% of observations	Censored observations	% censored observations
0 h	364 844	100.0	29 905	8.2
1 h	246 326	67.5	29 319	8.0
2 h	184 986	50.7	28 338	7.8
3 h	150 999	41.4	27 119	7.4
6 h	100 479	27.5	23 620	6.5
12 h	51 439	14.1	17 324	4.7
24 h	13 485	3.7	6 614	1.8
48 h	3 523	1.0	1 945	0.5
240 h	108	0.0	93	0.0

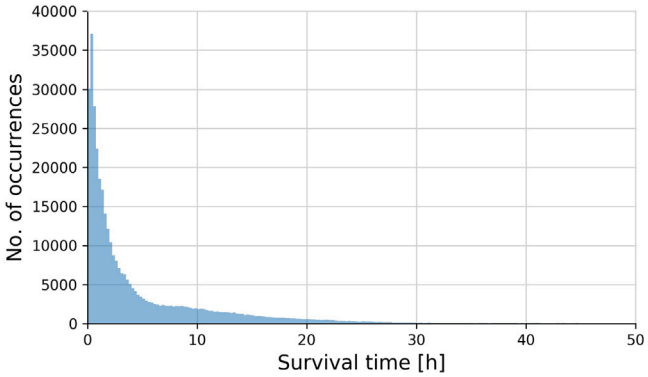


Fig. 3. Survival times distribution.

The data is heavily left-skewed, as around half of the pick-ups in the dataset happens in the first couple of hours after the drop-offs, as the data distribution in Table 2 shows. After 24 h, only 3.7 % of the cars still have not been picked up. The table also contains information on censored observations, which are described in the sub-section below.

Besides the car-sharing data, other features are added through feature engineering and from other data sources, such as OpenStreetMap (OpenStreetMap contributors, 2017), the DTU Climate Station data (Technical University of Denmark, 2019), BikeRoads data (Halldórsdóttir, 2015) and the Danish National Travel Survey (TU), which will be described in the following section. The elaborated features in the final dataset used for model training are shown in Table 3. All categorical variables are one-hot encoded, thus creating a lot of extra dimensions. Features obtained from the TU data are computed based on a representative sample of the population. We took respondents' answers and aggregated them per zone.

3.1. Censoring

As an objective with shared mobility systems is to maximise fleet usage, vehicles can be relocated from their current drop-off locations to more attractive locations if their estimated vacant time on the current location is high. This can be easily detected from the data, as pick-up location of one trip is different from the drop-off location of the previous trip the car had. Since from the data we had it is unknown at what time the car has been relocated, or how long the relocation process lasted, we assume the car is vacant at the drop-off location until its next pick-up time from the data. We introduce a small error here, however it should not be significant as the relocations were mostly done for the cars expected to be vacant for many hours, hence the relocation duration will not significantly impact these values, as it is usually much shorter. These observations will be treated as censored, as we know

Table 3
Features in the final dataset.

Variable	Type	Domain	Mean/Mode	Standard deviation
Survival time (T)	Numeric	≥ 0	5.99 h	14.45 h
Event indicator (E)	Categorical	0, 1	1	
Zone ID	Categorical	[0, 97]	8425	
Battery level	Numeric	[0; 100]	50.52%	24.24%
Hour	Categorical	[0, 23]	16	
Day of week	Categorical	[0, 6]	5 (Saturday)	
Air temperature	Numeric	[−10, 25]	6.67 C	6.50 C
Wind speed	Numeric	[0, 9]	2.69 m/s	1.65 m/s
Rainfall	Numeric	≥ 0	0.01 mm	0.02 mm
Distance from closest station	Numeric	≥ 0	0.75 km	0.61 km
Closest station type	Categorical	Train, metro, both	Train	
Distance from the city centre	Numeric	≥ 0	3.71 km	2.73 km
Area type	Categorical	[0, 5]	5 (High buildings)	
Zone — population (sample)	Numeric	≥ 0	N/A	N/A
Zone — avg. income	Numeric	≥ 0	N/A	N/A
Zone — avg. car sharing users	Numeric	0–1	0.01	0.01
Zone — avg. distance to work	Numeric	≥ 0	9.68 km	3.44 km
Zone — avg. bicycle ownership	Numeric	0–1	0.80	0.07
Zone — avg. PT commute ticket possession	Numeric	0–1	0.24	0.06
Zone — avg. household car ownership	Numeric	0–1	0.36	0.24

that the car was parked in a specific zone and was vacant until the point it was relocated, but it is unknown for how long it would have stayed parked had it not been relocated. We can see from Table 2 that 8.2 % of the data are censored data.

4. Exploratory analysis and general insights

We investigate here the significance individual features have on survival times. Their average effect is typically estimated using the Kaplan–Meier (KM) model, which is used to compute average survival functions for each features values, which then allows for easy comparison. Section 4.1 presents the location analysis, to see the spatial distribution of pick-ups, whereas the remaining sections analyse survival functions. The survival times used in these analyses include values of less than or equal to 50 h, which comprise more than 99 % of the data, hence all the survival functions converge almost to zero.

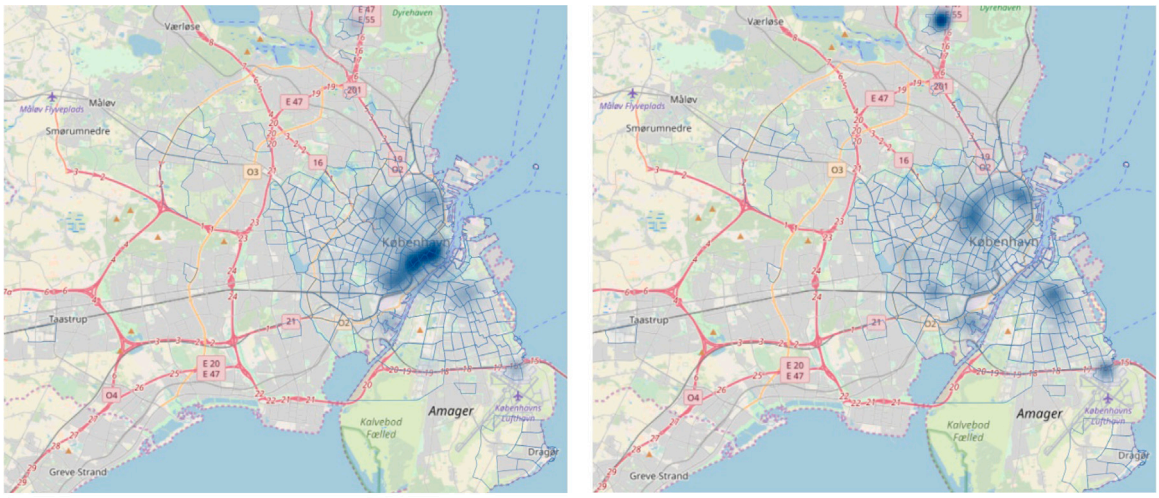
The Kaplan–Meier survival plots below are presented as follows: the x-axis represents time, starting from 0, which represents the instant when the car is dropped off, the y-axis represents survival probability, ranging from 0 to 1. For example, a survival probability of 0.8 at time 3 h means that on average there is 80 % chance that the car will be vacant for more than 3 h after being dropped off. Extending to the population level, it means that 80 % of the cars will be vacant for more than 3 h. In terms of the gradient, the steeper the curve the most likely is the pick-up will happen. We also applied Kolmogorov–Smirnov test to compute p-values in cases of two distributions.

4.1. Location effect

The simplest approach to predict survival times of cars is to use only vehicle locations as a predictor variable. For instance, Figs. 4a and 4b show the density of pick-ups of cars parked for less than and more than 10 h, respectively. The location effect can be clearly observed. For shorter survival times, the cars are often located in the city centre, whereas the longer survival times are spread out across the greater Copenhagen area. Therefore, some of the variance in the data can certainly be explained by using location variable only. Naturally, it is expected that other variables, such as time of the day and weather, also have an effect. We choose to show the plots for a 10-hour limit, however this can be used by a fleet operator to have an insight for any given hour the operator finds important from the business perspective. With the uncertainty of predictions, one can choose to use these historical data to position their vehicles or adjust pricing strategy.

4.2. Battery level

As the vehicle fleet consists of electric cars, we inspect the effect that the battery level has on the probability of the car being picked up. The battery level is represented as a percentage value, from 0 to 100, at the drop-off time. Fig. 5 shows the difference in survival times based on the battery level, with a 50 % battery level threshold. The p-value is 1.0. However, it can be seen that the cars with more than half of their battery level have lower survival times, i.e. are picked up faster. This is true throughout the timeline. The difference is quite significant, leading to the conclusion that making sure that cars have charged batteries should be a desirable operation strategy.



(a) Survival times less than or equal to 10 hours.

(b) Survival times greater than 10 hours.

Fig. 4. Density map of pick-ups based on survival times.

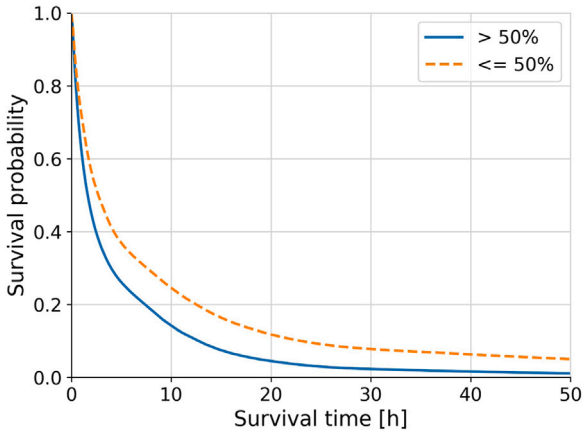


Fig. 5. Kaplan–Meier plot for the battery level.

4.3. Time

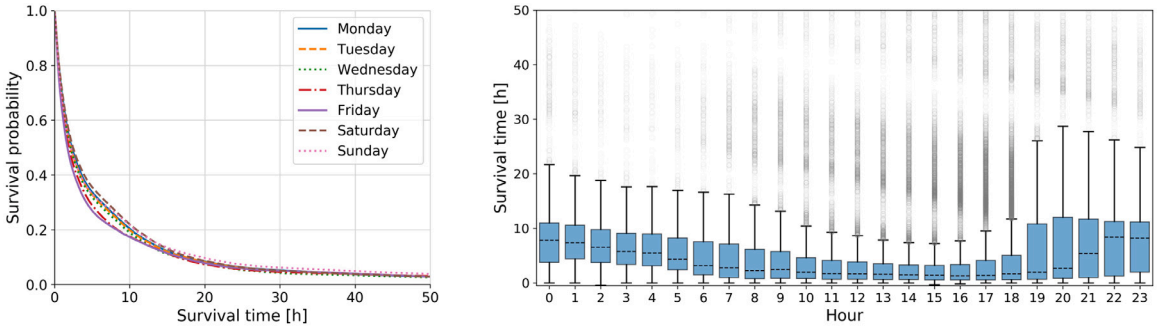
Time is expected to have a major influence on the survival times of cars. We can distinguish two time components: day of week and hour of day.

Day of week is used to capture daily patterns, such as working days and weekend. The KM plot for the day of week is shown in Fig. 6a. It is hard to spot major differences in this plot, with Friday and Saturday having lower survival times than the rest of the days.

Hour of day is expected to capture within-day variabilities, such as going to work/coming back from work and other activities. The effect of hour of day is shown in Fig. 6b, with several characteristic points for each hour. The median (or second quartile, i.e. Q2) survival time is plotted as the horizontal dashed line inside the rectangle, together with the 25 % and 75 % quantile (i.e. first and third quartile, Q1 and Q3), plotted as bottom and top rectangle sides, where the area between the two is referred to as the interquartile range (IQR). Lower and upper whiskers are bounded by $Q1 - 1.5IQR$ and $Q3 + 1.5IQR$, respectively. The observations outside this interval, usually considered as outliers, are plotted as circles.

The hour at which the cars are dropped off has a clear effect on survival times distribution. The cars are being used mostly during the working hours period, from 10 am up to 6 pm. Most of these trips are commuting and business-related trips. Cars dropped off after 8 pm show a rapid increase in the median and Q3 survival times, which then steadily decreases until the morning hours. This is a clear indicator for the relocation strategy, which should provide larger gains if done in off-peak hours.

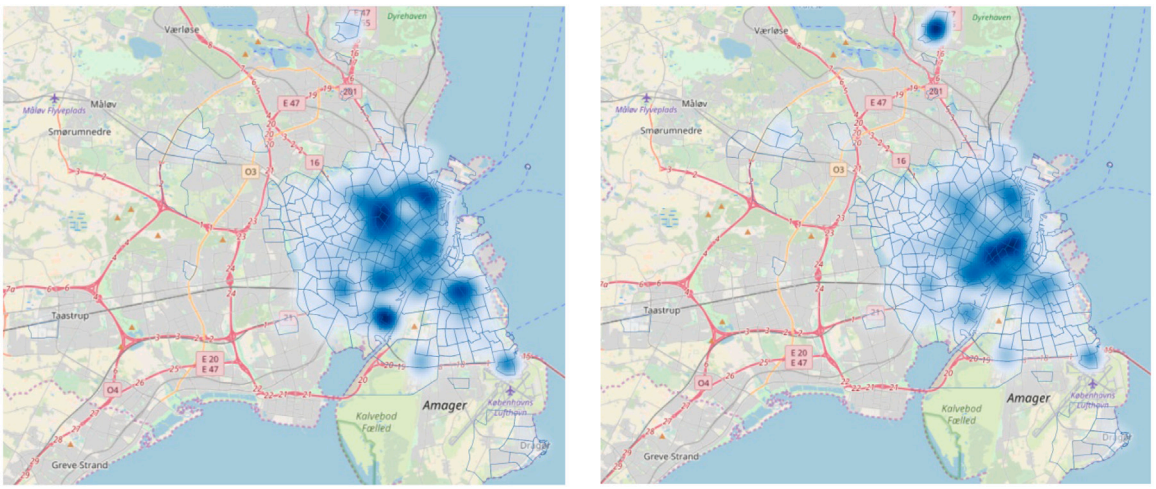
Fig. 7 shows spatio-temporal differences in pick-ups between morning and afternoon peak periods. In the morning, it seems that people from many areas of the city use the cars. In the afternoon, the busy areas are very specific. It is primarily the city centre and the university area in the north.



(a) Kaplan-Meier plot for day of week.

(b) Within-day variabilities in survival times.

Fig. 6. Time effect.



(a) Morning peak period: 07:00−09:00.

(b) Afternoon peak period: 15:00−17:00.

Fig. 7. Distribution of pick-ups based on time of day.

4.4. Weather

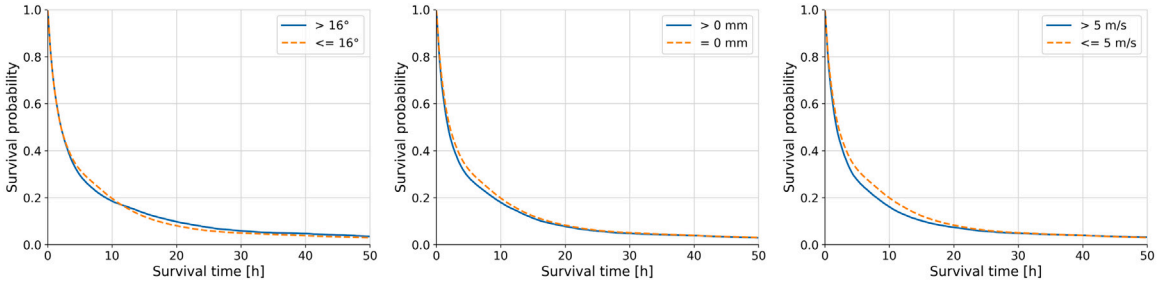
As one of the most bicycle-friendly cities in the world, many people in Copenhagen use bicycles on a daily basis, often also for commuting. It is expected that the weather influence cyclists choice to opt for using cars under prevalent poor weather conditions. To investigate this, weather data is collected for Kgs. Lyngby, Denmark, available from DTU Climate Station (Technical University of Denmark, 2019), which provides hourly measurements for air temperature, wind and rain. Using KM plots, we compare the temperature, rain, and wind above and below the 90 % quantile values. Difference in survival curves suggest that there is no large influence of weather on survival times. However, their corresponding p-value are 0.00.

Intuitively, lower air temperature should assume lower survival times, as cold weather will likely make people use cars instead of bicycles. The 90 % quantile of temperature is 15.7 degrees Celsius. The results are shown in Fig. 8a. The two lines are very similar, hence no significant claim can be made about the effect of air temperature.

Regarding the rainfall, it would be expected that a car parked when it is raining is picked up faster due to the poor weather. The results are shown in Fig. 8b. However, the rainfall data is not very precise, as it is measured for a specific area of the greater Copenhagen. As the city covers a large area, weather conditions may differ across the city. To account for this effect, a moving sum for a window of the past 6 h was calculated as the rain variable. A small difference in survival curves can be noticed, which support intuitive expectation that the cars are picked up a little bit faster when it is raining.

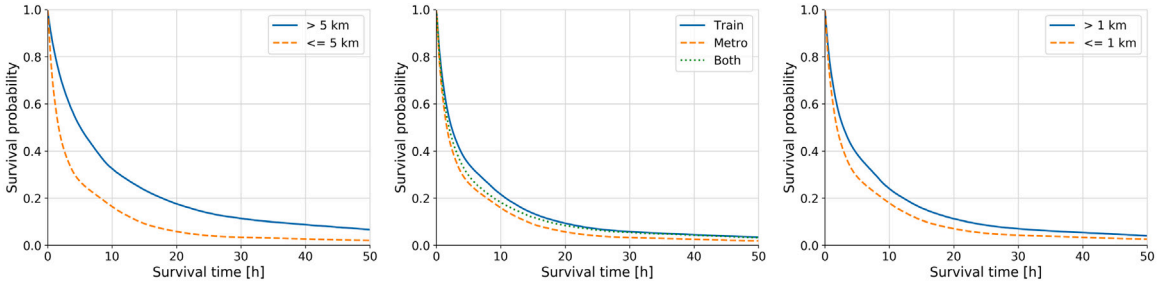
Lastly, the average hourly wind speed was investigated in Fig. 8c. In the plot, the solid line drops faster than the dashed one, meaning that when it is very windy the cars are picked up faster, with the strongest effect from all weather variables.

From these three analyses, it can be concluded that as the weather gets worse, the cars are used little bit more frequently. However, the exact quantification of the effect of weather on the survival times of cars would require the collection of precise location-wise hourly data. The weather data describe the weather at the hour when the car is dropped off, but the values can change over time. This might also be another reason why the rain effect cannot be found significant.



(a) Air temperature. (b) Rainfall. (c) Wind speed.

Fig. 8. The effect of weather on survival times.



(a) Distance to the city centre. (b) Distance per station type. (c) Distance to the nearest station.

Fig. 9. The effect of distance to transport hubs on survival times.

4.5. Distances to transport hubs

The region of the city where the cars are allowed to be parked covers a large area, with varying density of activities and public transport accessibility. Here we investigate the effect that the car location has in terms of the distance to the city centre (we chose the Nørreport station, which is one of the busiest stations), and the distance to the nearest train or metro station. Public transport is a quite attractive means of transportation in Copenhagen, where train and metro lines play a significant role in the system. It is a dominant option for both travelling within the larger city centre, as well as a main connection with the suburbs and surrounding cities. Therefore, distances to the stations are assumed to have an effect on possibly making multi-modal trips.

Fig. 9a depicts the difference between cars within the 5 km radius around the city centre and outside the 5 km radius. A significant difference can be observed, meaning that cars closer to the city centre are picked up much faster than cars further away. The p-value is 1.0. In Fig. 9b, different types of stations are compared: city-line S-train stations, metro stations and the stations being joint train and metro stations. We can observe that if the nearest station is a metro station then the cars are picked up faster than if the closest station is only an S-train station. Cars closer to train or metro stations are more active due to higher activity in these areas. If it is of both types, the survival is in between. Finally, the distance to the closest station was measured to not only have the information about which type of station is nearest, but also how near it is. In Fig. 9c, the cars closer than 1 km are compared to those further away from a station. The p-value here is 1.0. This can directly impact the locations when rebalancing vehicles, with central zones and areas close to the stations being more desirable destinations.

4.6. Area type

From the BikeRoads dataset provided by DTU, all the roads in Copenhagen are assigned one out of six different area types. The city centre mainly consists of mixed settlement roads. When moving outside the centre, the area type is mainly low settlement. Roads classified as nature, parks, high settlement and industry are more spread out and not as frequent. To include this information, for each parked car we find its closest road and assign the corresponding road category.

Fig. 10 shows the KM plot for all six different area types. It is not trivial to spot clear distinctions, however some interesting conclusions can be drawn. Areas with buildings (high and low) have the longest survival times, which may be explained by the fact that cars stay parked overnight. Nature and mixed areas have the lowest survival times. The values for mixed areas intuitively make sense as these roads were seen to be located mostly in the centre of Copenhagen. The values for nature are more surprising, but could be due to the fact that people use the car both to go to and come back from a nature area. Therefore, cars will not be parked in those areas for very long. Industry and park and sport are the average among all.

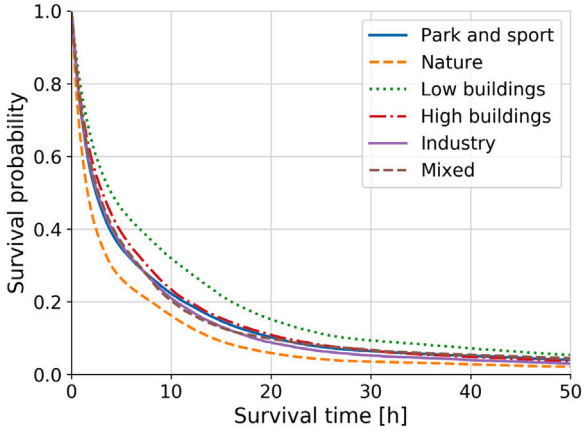


Fig. 10. Kaplan–Meier plot for area type.

4.7. Conclusions

Although these results look at average observed behaviour, as a general insight the conclusions can still be very valuable. It is interesting to notice that at the initial hours in most of the KM plots, the curves appear to be very close, almost overlapping. This can be explained by the fact that shorter survival times are more random than longer.

5. Results and comparison of survival modelling approaches

5.1. Experimental setup

In this section, we test and compare linear and neural-network-based survival models, CPH and DeepSurv, respectively, in predicting survival time distribution. Two approaches were investigated:

- A one-step approach, where we apply survival analysis on the whole dataset, as depicted in Fig. 1a.
- A two-step approach, where we first classify survival times based on a certain threshold and then apply survival analysis on the two classified groups of data observations, as depicted in Fig. 1b. The two-step approach contains classification error, therefore we also evaluate the survival predictions on the true split (i.e. in the case of perfect classification), where we manually split survival times of the cars on the specific threshold and then run survival analysis on those splits. In this way, we can see the effect of the classification error on the final prediction accuracy.

For each approach, three models were investigated:

- "CPH baseline": the baseline model is the CPH model using only zone ID as a single feature for prediction. This model should indicate if there is a need for including more predictor variables or a simple zone-based prediction model can provide sufficiently good results.
- "CPH": the CPH model using all the features. Being a linear model, this model will be used for direct comparison with DeepSurv, a non-linear model described below, to estimate how much a model's predictive capability can be improved by exploiting non-linear relationships in the data.
- "DeepSurv": a deep neural network model using all the features. This is the most complex model and our target model for validation of the application of deep survival modelling in shared mobility data.

Therefore, we had to train an overall of 99 models: 3 models for the one-step approach and 96 models for the two-step approach (the combinations of 3 survival models, 8 thresholds, 2 models per threshold and 2 split datasets (predicted and true split)). Out of 99 models 33 were DeepSurv models, where we needed to do the hyper-parameter tuning for each one of them separately. In addition, for the classification task, we needed to train 24 models, where hyper-parameter tuning was also required. Overall, this was very computationally intensive and time-consuming procedure that required on average about 9 h per model on a machine with an AMD Ryzen ThreadRipper 2990WX 32-core CPU, 64 GB of RAM and using one NVIDIA GeForce GTX 1080 Ti GPU. All the experiments were done in Python (Rossum and Drake, 2011), using libraries for data manipulation (McKinney), numerical computing (Walt et al., 2011), scientific computing (Virtanen et al., 2020), machine learning (Pedregosa et al., 2011; Abadi et al., 2015; Chollet et al., 2015; Chen and Guestrin, 2016) and plotting (Hunter, 2007).

The dataset was divided into training and test sets. The training set comprises 80 % of the data and the test set has the rest 20 %. To find the best hyper-parameters for each model, a grid search was used. All categorical features were one-hot encoded and all

Table 4
The one-step approach evaluation results.

Model	C-index
CPH baseline	0.601
CPH	0.658
DeepSurv	**0.683**

numerical features were standardised to have zero mean and unit standard deviation. To evaluate the performance of the survival models, the C-index was used, described in Section 2.2.4, which calculates how well the models predict the order in which the cars will be picked up.

As previously seen from Table 2, a big percentage of the observed pick-up times lies in the initial few hours after a car has been dropped off. From the exploratory Kaplan–Meier plots in Section 4, it can be seen that most features do not separate the data very well in the initial hours. This is also a reason for applying classification, expecting that the separation of shorter and longer survival times can make the survival models perform better.

5.2. One-step approach

First, the simplest model was fitted, which is the CPH baseline to check if the zone ID feature alone carries enough information for prediction. Then, CPH and DeepSurv models were trained on data using all the features. According to the hyper-parameter tuning results, for CPH baseline and CPH, L2 regularisation of 0.2 was selected. DeepSurv model consists of 2 hidden layers, with 64 and 32 nodes respectively, ReLU activation function, kernel regularisation of 0.01 and the dropout rate of 0.3. Adam optimisation algorithm was used for training with a learning rate of 0.001. Table 4 presents the concordance indices for the three approaches.

CPH baseline achieved a C-index value of 0.601, whereas CPH and DeepSurv outperformed it, indicating that using only zone information to predict survival times is not enough to provide accurate predictions. CPH achieved a C-index value of 0.658 and DeepSurv a value of 0.683. DeepSurv provides better accuracy, however the difference is not very large (2.5 %). This is an indicator that there are no significant non-linearities in the data that a neural network would benefit from comparing to a linear model. The reason DeepSurv does not demonstrate more superior performance may be a limited number of features: the explanatory power of the features we were able to provide may not be sufficient to explain the variance in the data.

5.3. Two-step approach

One of the arguments for implementing the two-step approach was the situation that the cars can often end up in one of two scenarios. First, a car is dropped off in a good location and will stay vacant for a short time during the day, or at most overnight to be then picked up the next morning. Second scenario is that the car has been left in a less active location, where the average vacant time is high. We therefore introduce classification of vehicle vacant times, however not in the extreme way as mentioned above but in a more flexible way by specifying a threshold and try to discriminate between these two cases. This also allows a car fleet operator to chose a threshold based on their business interest and make relocations accordingly.

5.3.1. Classification results

To predict observations above and below the split time, a classification model was created with two different classes: class 1 for survival times $T \leq t$ and class 0 for survival times $T > t$. Four models were tested:

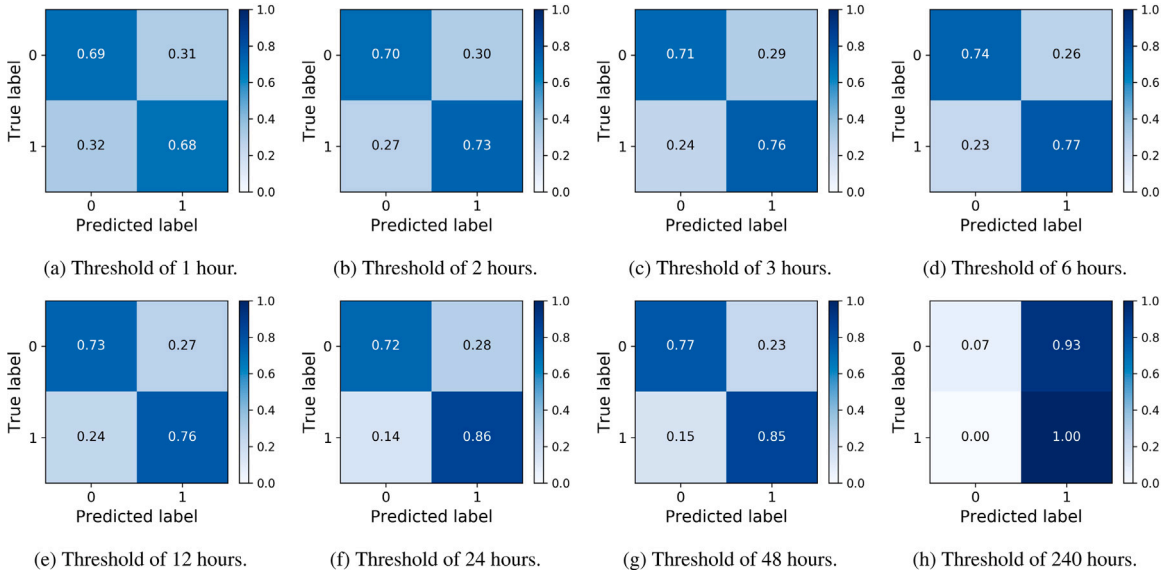
- Baseline model, a dummy classifier making predictions based on simple rules — respecting the training set class distribution.
- Logistic regression (LR), as the simplest yet often powerful model.
- Multi-layer perceptron (MLP), a neural network with 1 hidden layer with 100 nodes and a logistic activation function.
- Gradient-boosted decision trees (GBDT), as a widely used model which often shows the best performance.

Training data was separated based on the threshold, where for shorter survival times only uncensored data were used. To address class imbalance, class weights were used to scale up minor class. We tested several thresholds to see if there is an optimal split that provides the best results. The thresholds tested were 1, 2, 3, 6, 12, 24, 48 and 240 h. The split at 1 h divides the dataset into similar sizes of about 50 % each, and as the split increases the first split contains more data, with the split at 48 h the first split contains about 99 % of the data, and the split at 240 h the first split contains almost the whole dataset (see also Table 2).

Results of the classification are shown in Table 5. In general, the three models performed with satisfying results, outperforming the baseline model, with an average F_1 score of over 0.7. More specifically, the LR model showed inferior performance, the MLP model demonstrated very good performance, at some places very close and identical to GBDT, and the GBDT model performed the best with the highest F_1 scores. All the models, and especially GBDT, do a very good job in classifying the observations, which was an encouraging result we hoped for. For more detailed insights, confusion matrices for the best performing classification model for each threshold are shown in Fig. 11. They show the percentage of correctly and wrongly classified samples for each class, where the perfect results would be 1s on the main diagonal and 0s otherwise. The class imbalance problem seems to be handled properly, as the values on the main diagonal are similar in magnitude. The exception is the threshold of 240 h, where most of the data belongs to class 1 and only a handful of observations to class 0, thus the score of only 0.07 percent correctly classified observations of class 0. As GBDT performed best, its results will be used as input to the survival analysis step.

Table 5
Classification results for all thresholds and all machine learning models.

Threshold [h]	F_1 score			
	Baseline	LR	MLP	GBDT
1	0.419	0.683	0.467	**0.702**
2	0.522	0.705	0.730	**0.735**
3	0.574	0.723	**0.765**	**0.765**
6	0.647	0.756	0.785	**0.790**
12	0.704	0.745	0.778	**0.785**
24	0.821	0.811	0.849	**0.871**
48	0.762	0.822	0.784	**0.854**
240	**0.989**	0.860	**0.989**	**0.989**



(a) Threshold of 1 hour. (b) Threshold of 2 hours. (c) Threshold of 3 hours. (d) Threshold of 6 hours.

(e) Threshold of 12 hours. (f) Threshold of 24 hours. (g) Threshold of 48 hours. (h) Threshold of 240 hours.

Fig. 11. Confusion matrices for the best performing classification model (GBDT) for each threshold.

5.3.2. Survival analysis results

Trained classification models can make predictions on whether an observation belongs to a class of shorter or longer survival time for a defined threshold. By making predictions on the test set, it now creates two smaller datasets. Two separate survival models are trained for each dataset separately. In order to compare it to the one-step approach, we need to calculate the C-index on the complete test set. We do this by computing survival functions for all test set observations, calculating median survival times for each observation and use these values to calculate C-index. Using predicted risk or hazard values may lead to biased results, as two models may have different scales, since the risk values are on an arbitrary scale, i.e. they are not used for interpretation of their absolute values, but to compare values among the samples to determine the ranking. Thus, combining values from two different models must be done on a comparable metric, for which we chose the median survival time. The results for each threshold and model, evaluated both separately and jointly, are shown in Table 6.

The graphical representation of the joint evaluation, together with the one-step approach results, is shown in Fig. 12. The results of the one-step approach start with "1" in the legend and are depicted using dashed lines. The lines are flat as this approach results in only one value (regardless of the split threshold) which is plotted together with the other results for easier comparison. The results of the two-step approach start with "2" in the legend and are depicted using solid lines. Model evaluation on the true split is denoted with "true split". "CPH baseline" model is depicted in green with no markers, "CPH" is depicted in orange with the "X" marker and "DeepSurv" is depicted in blue with the circle marker.

We tested various split thresholds, shown on the x-axis, whereas on the y-axis is the C-index, which is our accuracy measure. We can see that for shorter thresholds we can obtain much higher accuracy with the two-step approach, and as the threshold increases, i.e. more data belong to one class, the two-step approach converges to the one-step approach, as expected. For the split at 1 h, i.e. where we classify the idle times of cars to be less than or greater than 1 h, and then run DeepSurv, we obtain the most accurate prediction. The overall effect of adding the classification and making it a two-step approach is clear. An increase in C-index values of up to 15 % is achieved.

To get an intuition of how well predicted survival functions correlate with observed survival times, we plot them together for comparison. In Fig. 13, we can see examples of three randomly chosen uncensored observations from the dataset with their

Table 6
The two-step approach C-index evaluation.

Threshold [h]	Model	Split data	
		True	Predicted
1	CPH baseline	0.775	0.760
	CPH	0.801	0.765
	DeepSurv	0.809	**0.804**
2	CPH baseline	0.776	0.767
	CPH	0.797	0.767
	DeepSurv	0.801	**0.792**
3	CPH baseline	0.761	0.754
	CPH	0.782	0.761
	DeepSurv	0.787	**0.774**
6	CPH baseline	0.724	0.722
	CPH	0.752	0.734
	DeepSurv	0.765	**0.753**
12	CPH baseline	0.663	0.660
	CPH	0.713	0.698
	DeepSurv	0.730	**0.721**
24	CPH baseline	0.612	0.604
	CPH	0.668	0.663
	DeepSurv	0.690	**0.686**
48	CPH baseline	0.603	0.596
	CPH	0.661	0.659
	DeepSurv	0.685	**0.683**
240	CPH baseline	0.601	0.601
	CPH	0.659	0.659
	DeepSurv	0.684	**0.684**

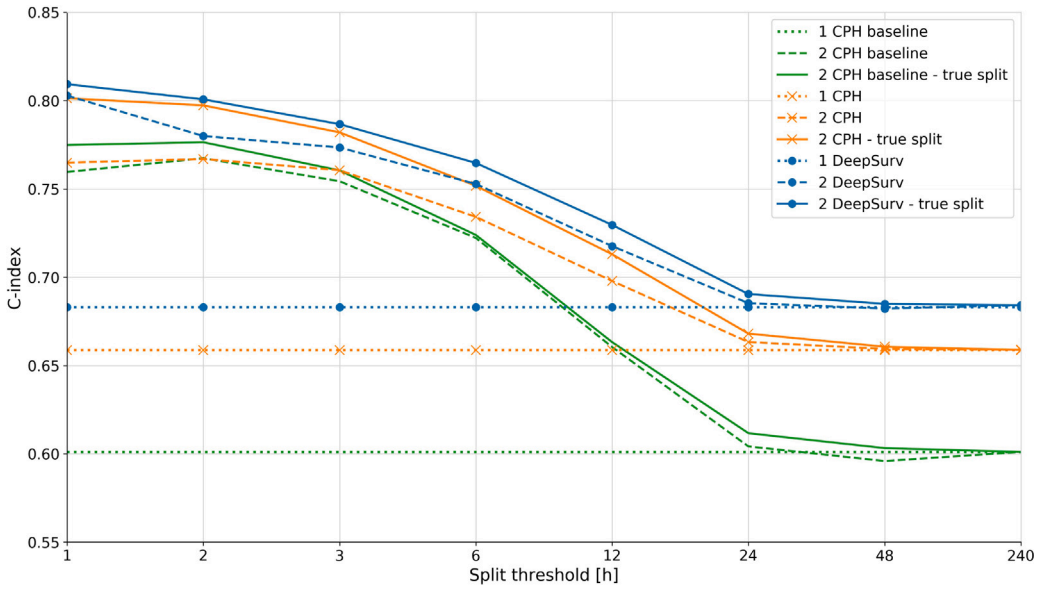
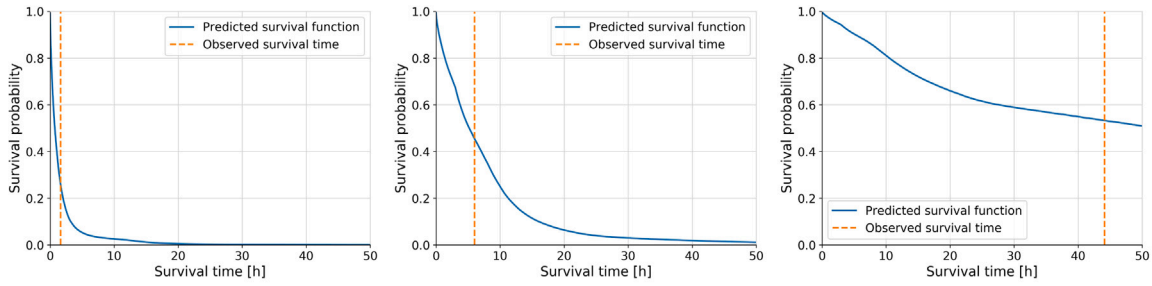


Fig. 12. The results of one-step and two-step approaches for three models.

observed survival times and the corresponding predicted survival functions using the two-step approach for the split of 3 h. We chose the observations to be uncensored as we need true survival times to be known for this comparison. We can observe that the survival functions have a steeper descent around the observed survival times, meaning that our model predicts that there is a higher probability of the event happening in those periods. This behaviour is especially obvious for the first two examples and is less explicit in the last example due to the significantly less amount of data with large survival times present in the dataset. We can see that there is uncertainty in the data, which we capture using survival analysis. This is exactly the behaviour we wanted to achieve with our model. By including more and better data in terms of precision and new features, we can expect that the uncertainty decreases, with survival curves being even steeper around observed survival times and flatter in other periods.



(a) Observation 1: observed survival time 01:40:00 h

(b) Observation 2: observed survival time 06.01:00 h

(c) Observation 3: observed survival time 44:07:00 h

Fig. 13. The comparison between predicted survival functions and observed survival times for three randomly chosen uncensored observations from the dataset.

5.4. The application of Gaussian copulas

To refine individual survival functions through the correlation with neighbouring vehicles, we apply Gaussian copulas. In a general case, a copula can encapsulate as many vehicles as needed, even all the vacant vehicles in a shared mobility system. This is however not necessary, as most of the vehicles will be distant enough to result in a zero correlation between them, hence it will only increase computational requirements without the modelling effect. For a given vehicle, the number of vehicles included in its copula will typically include all vacant vehicles within a defined radius. The vehicles outside the radius will be assumed to have zero correlation with the given vehicle. The copula is constructed using the Gaussian kernel, i.e. the radial basis function (RBF) kernel (also called standard exponential kernel), computed using the direct distance between vehicles in metres. The RBF kernel is expressed as:

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right) \tag{12}$$

It has two parameters: the length scale l, which means that our function cannot extrapolate beyond this value. We set it to 300 metres (about 984 ft), meaning that vehicles at a distance of more than 300 metres will not be correlated, i.e. their survival functions will be independent. The second parameter is the output variance σ^2, which is the average distance of the function from the mean. As this is just a scale parameter, we set it to its default value of 1.

When computing the values for the kernel, as the Euclidean distance is always positive, we explicitly place a sign to obtain positive or negative correlation. The positive correlation means that the change of one vehicle will increase the survival function of the other vehicles. This happens when a vehicle is picked up, consequently making close by vehicles to have a lower chance of being picked up from that moment on. On the contrary, when a vehicle is dropped off, a copula is dynamically created or extended to accommodate the new vehicle, and the survival functions are negatively affected, i.e. decreased, meaning that on average every vehicle have a higher chance of being picked up. The effect of the influencing factors such as time of day, zone centrality, weather conditions, etc. are captured in the marginal survival functions, and the copula aims at modifying those functions.

We can see in Fig. 14 an example of a Gaussian copula applied in the context of car sharing with two and three vehicles, i.e. we have a multivariate copula with two (Scenario 1) and three (Scenario 2 and 3) variables respectively, representing the correlation between the vehicles. In this way, as we fix the value of one variable, we get a conditional distribution on the other variables.

In Fig. 15 we can see a correlation effect that copula produces. Let us assume that there is a vehicle dropped off at a certain instant in time, and its survival function starts at that instant in absolute timeline. The function for correlation 0 can be considered a baseline, i.e. an independently estimated survival function with no impact of the other vehicle. We now test the sensitivity effect the other vehicle has on it. In the case of negative correlation, we assume a second vehicle in its vicinity is being picked up after 3 and 7 h (graphs left and right, respectively). We can observe that for highly correlated vehicles (i.e. vehicles are very close to each other), the impact is the highest. In the case of positive correlations, we assume the second vehicle is being dropped off after 3 and 7 h. Similarly, higher correlations have higher effect on the survival function. By comparing two graphs, we can also observe that the correlation effect is higher when the effect happen earlier (after 3 h instead of after 7 h).

6. Comparison with other models

To be able to conduct a comparison between a survival model and typical non-censored regression models, we must assume an approximation of the survival prediction. As regression prediction is a point estimate, we must provide an equivalent quantity from the survival distribution prediction, which is typically median or mean survival time.

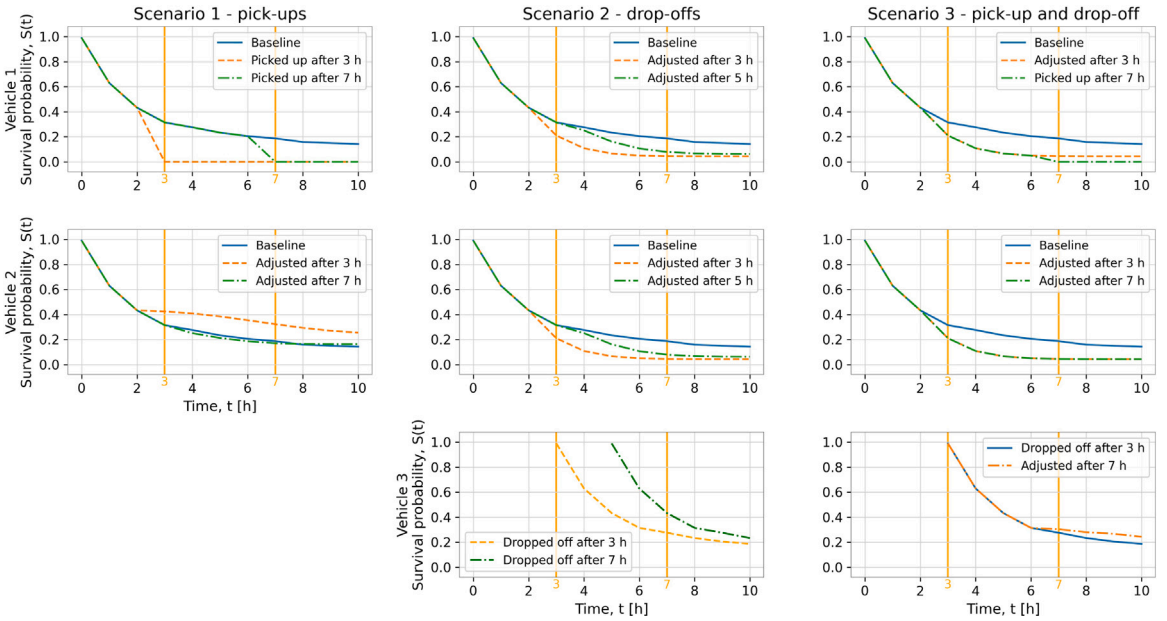


Fig. 14. An example of the correlation effect with three scenarios. Scenario 1 — the effect of the vehicle 1 pick-up on vehicle 2 with two example cases of the event time (picked up after 3 and 7 h). Scenario 2 — the effect of the vehicle 3 drop-off on vehicles 1 and 2 with two example cases of the event time (dropped off after 3 and 7 h). Scenario 3 — the effect of the vehicle 3 drop-off and subsequent vehicle 1 pick-up event (after 3 and 7 h, respectively) on other vehicles.

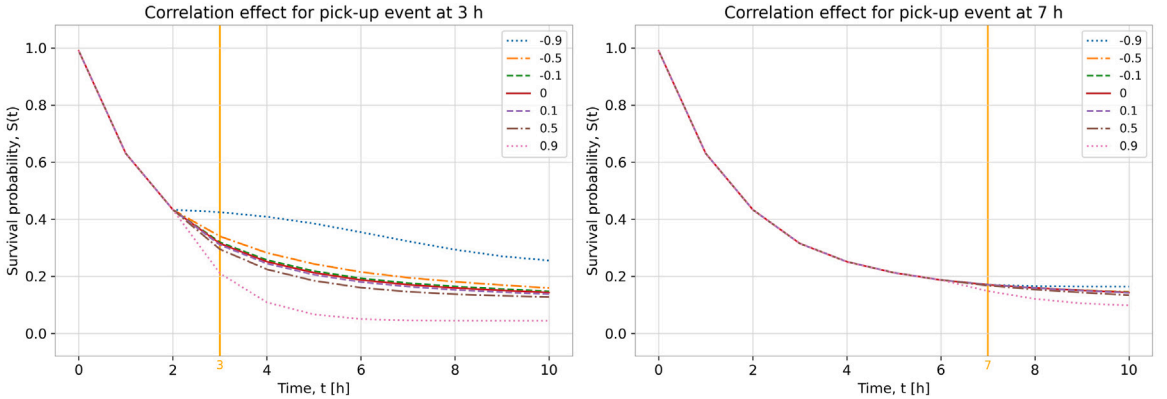
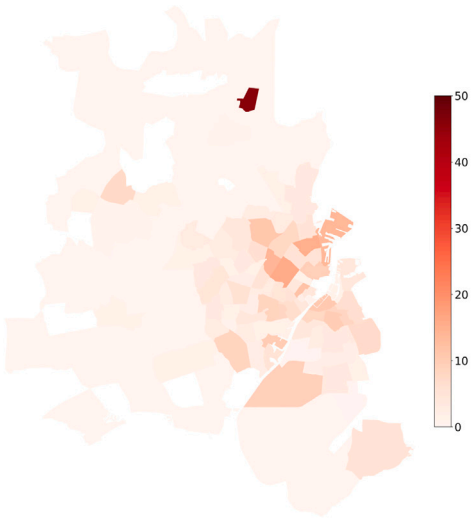


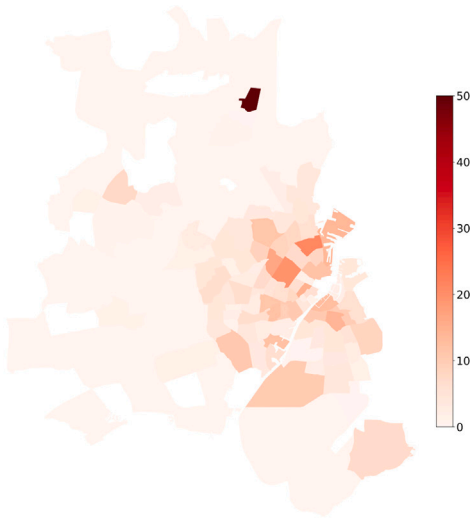
Fig. 15. An example of the correlation intensity effect.

6.1. Comparison with a neural network regression model

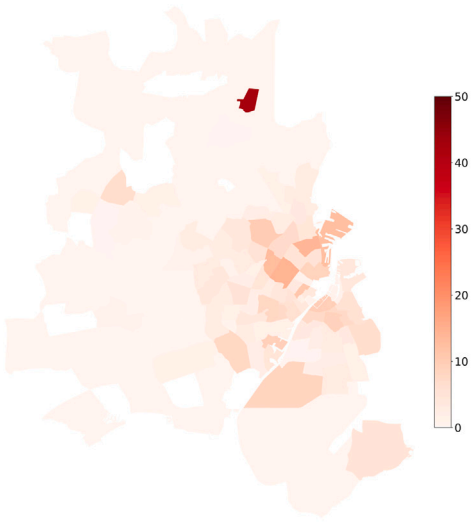
Here we trained a multi-layer perceptron (MLP) neural network with a similar configuration as the DeepSurv neural network has: two hidden layers with 128 and 64 nodes respectively, ReLU activation function and L2 kernel regularisation of 0.01, optimised using the Adam optimisation algorithm with a learning rate of 0.001. For DeepSurv, we compute median survival time for comparison, which is the same quantity we use for computing C-indexes. We provide two comparisons for better insights: the mean squared error (MSE), typical for a standard regression problem, and C-index, typical for survival analysis. The tests show that in both cases survival model outperforms the MLP model. In terms of the C-index, where we use MLP model predictions to compute it, we obtain a C-index of 0.668. This is worse than DeepSurv, which reaches 0.804. With respect to MSE, we compute it using true and predicted vacancy times for all cars in the test set. Similarly, in this case as well DeepSurv outperforms MLP by a small margin, yielding MSE of 36.57, while MLP obtains a MSE of 37.23. We note that for computing MSE, we use data that are $\leq$ 48 h, which comprise more than 99 % of the data, to avoid bias caused by long vacancy times, and therefore contributing with large error. This is done because there is very few data points to fit the model and hence we do not expect good predictions in those cases, as it suffers from the epistemic uncertainty. Given that the error gap in C-indexes is larger that in MSE, we suspect that it is due to short vacant times, as



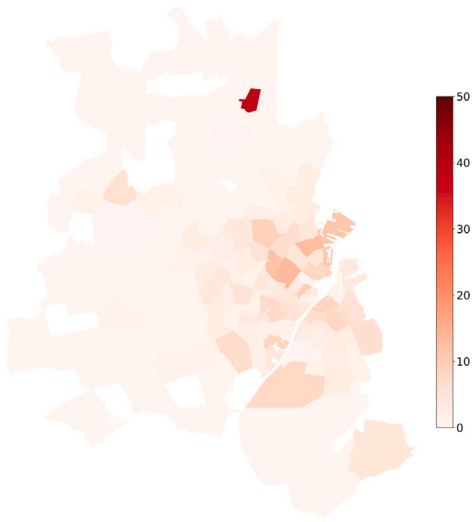
(a) Historical average baseline.



(b) SARIMA time series model.



(c) Multi-layer perceptron neural network regressor.



(d) DeepSurv survival model.

Fig. 16. Average MSE per zone for different prediction models.

Table 7
Comparison between MLP regressor and DeepSurv median survival time.

Model	MSE	C-index
MLP regressor	37.23	0.668
DeepSurv — median survival time	**36.57**	**0.804**

their contribution in MSE is smaller but it is where the survival model is able to predict much better values. The overview is shown in Table 7.

6.2. Zone-based predictions using time series models

To move from vehicle-based to aggregated prediction comparison, we run a zone-based time series prediction, to obtain another benchmark. Here we also need to obtain aggregated zone-based predictions from individual vehicle survival predictions. We do this

by making survival predictions for all vehicles individually, taking median survival time, computing the time until the vehicle will be vacant (the sum of the drop-off time and the median survival time) and aggregating by zones based on vehicles' locations. For the MLP neural network the procedure is similar as with survival predictions, with a difference that we use its predictions instead of the median survival time. In addition, we compute the historical average number of pick-ups per zone per time interval as the baseline.

There are 117 zones. We fit the time series model for each zone separately. We apply time discretisation as well, with six daily intervals (0–6 h, 6–9 h, 9–15 h, 15–18 h and 18–21 h and 21–24 h). We use the SARIMA model, which is an autoregressive model, where the prediction depend only on the previous time series values. We used last 85 intervals as test data, which is about a 2-week period.

The results are shown in Fig. 16. We can see that the zones with largest errors follow a similar pattern and are mainly located in the city centre area, which are the most active zone with high numbers of pick-ups and drop-offs, and with high variability in demand as well, which explains the largest errors. There is another zone in the north with high error, the DTU zone, which is a large origin and destination of trips. Summing up individual zones' mean squared errors we obtain total MSE per model: historical average with a value of 477 and SARIMA with a value of 568. MLP and DeepSurv model preform better, with the values of 412 and 403, respectively. This comparison is however not just, as for the latter two the drop-off time is correct by default, which makes the prediction time easier, hence it should be taken with reserve, yet it is the only way to compare two types of models.

7. Conclusion

In this paper, we investigated the application of survival models to the transport demand prediction problem. We successfully applied both traditional and deep survival analysis in the context of shared mobility systems. We also provide a method to account for spatial correlation among the vehicles. We demonstrated that these methods can be very informative and, therefore, provide predictions that can be used for optimising shared vehicle fleets. The CPH models demonstrated a strong fit on the car-sharing data, while DeepSurv was overall a stronger predictor than CPH.

We also proposed a two-step approach, where the survival times are first classified with respect to a threshold into the two classes, namely, with shorter and longer duration, before fitting survival models. The two-step approach outperforms the one-step approach for all survival models and threshold values. The results show that the two-step approach using DeepSurv was significantly better than the other models. It was clear from the classification models that the variables available in the data can separate the two scenarios fairly well. However, the two-step approach is more demanding to fit as it requires more hyper-parameter tuning.

Regarding data, getting access to the data describing exactly when cars are relocated could improve the predictive power of the models, since a more accurate measurement for censoring could be used. The weather data might have a greater influence on the survival times than seen in this paper. The weather was logged for that exact hour at which the car was dropped off, which does not take into account the time-dependent feature change. If a car has longer survival time, it should not be assumed that weather-related features, e.g. rainfall, has stayed constant during the time passed. More precise incorporation of time-dependent features should be considered in future work.

Future work can consider reducing the dimensionality of the input space. As the zone feature is one-hot encoded, it creates many dummy variables. This can be potentially improved through the use of embeddings, preferably explicitly including zone economic and demographic characteristics, such as population density and median income. In addition, a new application for these data and prediction models can look into the expansion of the operation area of shared mobility service. By using the dependencies with variables like area type, distance to the city centre and distance to the train station, it may be possible to predict how attractive a new area would be for business expansion.

CRediT authorship contribution statement

Bojan Kostic: Conceptualization, Investigation, Software, Writing - original draft, Writing - review & editing. **Mathilde Pryds Loft:** Software, Writing - original draft. **Filipe Rodrigues:** Conceptualization, Writing - review & editing, Supervision. **Stanislav S. Borysov:** Conceptualization, Writing - review & editing, Supervision.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement no. 754462. The authors would like to thank SHARE NOW Copenhagen for providing the data for this research.

Table A.1
Minimum cost flow relocation results.

From Zone ID	To Zone ID	No. of cars
10 211	10 212	1
10 233	10 222	1
10 234	10 222	3
10 235	10 222	1
10 242	10 214	2
10 244	10 213	1
10 244	10 214	1
10 251	10 241	2
10 285	10 217	1
10 285	10 218	4
10 285	10 281	3
10 324	10 317	1
10 324	10 319	1
10 326	10 319	1
14 712	10 216	1
14 712	14 711	1
14 713	10 216	1
14 713	10 281	1
18 512	18 520	1

Appendix. Minimum cost flow problem for the relocation of cars

We provide an experiment to show that our predicted survival times can provide reasonable results when it comes to the relocation of cars. The aim of this optimisation is not to provide a state-of-the art method for vehicle relocation optimisation, but to provide an experiment on the usability of vacancy time predictions. To optimise car vacancy time, we define the optimisation problem as a minimum cost flow problem. The objective is to minimise the total travel distance for cars relocation while moving cars from less attractive to more attractive zones, i.e. to balance the supply and demand. We introduce spatial discretisation in terms of zones, and use the geometric mean location of the parked cars as node representation for each zone. As we look at the problem as graph optimisation, zones are represented by nodes and connections between them by arcs.

The number of vehicles in a zone at a time period is considered supply, and demand is predicted. We do not take into account other costs, such as personnel cost. To apply the minimum cost flow model with a feasible solution, we need to address the misbalance between demand and supply and hence we introduce a dummy node with a zero access cost. This ensures that all demand will be met, as any extra supply can travel to this node with no additional cost and not affecting the objective function value. The dummy node has no outgoing arcs to prevent its traversal.

Our network is represented as a directed graph $G = (V, E)$, with set of vertices V and set of edges E. The relocation is done from an origin vertex $o \in V$ to a destination vertex $d \in V$, through the edges $(u, v) \in E$, with the constraints that edge capacity is positive, i.e. $c(u, v) > 0$, flow f is non-negative, i.e. $f(u, v) \geq 0$, and edge weight $w(u, v)$. The solution is to minimise the total cost of the flow over all edges:

$$\min \sum_{(u,v)\in E} w(u, v) f(u, v) \tag{13}$$

with the constraints in terms of capacity $f(u, v) \leq c(u, v)$, which we release in our case and allow for infinite capacity due to low number of vehicle relocations compared to link capacities.

For the minimum cost flow problem, we have 111 nodes. The travel distance between nodes is used as arc cost, which are obtained as typical travel time using Google Maps, for the relocation optimisation period. We choose a single time period from 18–21 h. Based on median vacancy time predictions for all vehicles, we compute the aggregated demand for each zone. Supply is also given by the vacant vehicles. There are 28 vehicles to be relocated, as shown in Table A.1. The results of the minimum cost flow problem show that the total cost of relocations is 57.37 km. Given the driving distances in Copenhagen, this seems like a very reasonable travel distance to relocate all the needed cars. For many cars, an optimal solution may be just to relocate them in an adjacent zone. This is a simple example that proves that survival prediction can be used for operational use cases of the fleet operators. Investigating more sophisticated relocation optimisation approaches and strategies is outside the scope of this paper.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Manjunath, Kudlur, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, Zheng, Xiaoqiang, 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/.

Almannaa, Mohammed H., Elhenawy, Mohammed, Rakha, Hesham A., 2020. Dynamic linear models to predict bike availability in a bike sharing system. Int. J. Sustain. Transp. 14 (3), 232–242.

Ashqar, Huthaifa I., Elhenawy, Mohammed, Almannaa, Mohammed H., Ghanem, Ahmed, Rakha, Hesham A., House, Leanna, 2017. Modeling bike availability in a bike-sharing system using machine learning. In: 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE, pp. 374–378.

Balac, Milos, Becker, Henrik, Ciari, Francesco, Axhausen, Kay W., 2019. Modeling competing free-floating carsharing operators – a case study for zurich, switzerland. Transp. Res. C 98, 101–117.

Balac, Milos, Ciari, Francesco, Axhausen, Kay W., 2017. Modeling the impact of parking price policy on free-floating carsharing: Case study for zurich, switzerland. Transp. Res. C 77, 207–225.

Becker, Henrik, Balac, Milos, Ciari, Francesco, Axhausen, Kay W., 2020. Assessing the welfare impacts of shared mobility and mobility as a service (maas). Transp. Res. Part A 131, 228–243.

Bellotti, T., Crook, J., 2009. Credit scoring with macroeconomic variables using survival analysis. J. Oper. Res. Soc. 60, 1699–1707.

Chen, Tianqi, Guestrin, Carlos, 2016. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16. ACM, New York, NY, pp. 785–794.

Chollet, François, et al., 2015. Keras. https://keras.io/.

Ciari, Francesco, Bock, Benno, Balmer, Michael, 2014. Modeling station-based and free-floating carsharing demand: Test case study for berlin. Transp. Res. Record: J. Transp. Res. Board 2416 (1), 37–47.

Cox, David Roxbee, 1972. Regression models and life-tables. J. R. Stat. Soc. Ser. B Stat. Methodol. 34 (2), 187–220.

2019. DriveNow. https://www.drive-now.com/dk/en/copenhagen. Online; accessed 28 2019.

Erhardt, Gregory D., Roy, Sneha, Cooper, Drew, Sana, Bhargava, Chen, Mei, Castiglione, Joe, 2019. Castiglione do transportation network companies decrease or increase congestion? Sci. Adv. 5 (5).

Finkelstein, Maxim, 2008. Failure Rate Modelling for Reliability and Risk. In: Springer Series in Reliability Engineering, Springer, London, UK.

Friedman, Jerome H., 1999. Greedy Function Approximation: A Gradient Boosting Machine. Technical Report, Stanford University, Stanford, CA.

Gammelli, Daniele, Peled, Inon, Rodrigues, Filipe, Pacino, Dario, Kurtaran, Haci A., Pereira, FranciscoCamara, 2020. Estimating latent demand of shared mobility through censored Gaussian processes. arXiv e-prints, arXiv:2001.07402 [stat.ML].

Halldórsdóttir, Katrín, 2015. Behavioural Models for Cycling - Case Studies of the Copenhagen Region (Ph.D. thesis). Technical University of Denmark, Kgs. Lyngby, Denmark.

Harrell, Jr., Frank E., 2015. Regression Modeling Strategies, second ed. In: Springer Series in Statistics., Springer, Switzerland.

Hastie, Trevor, Tibshirani, Robert, Friedman, Jerome, 2017. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed. In: Springer Series in Statistics., Springer, New York, NY.

Hunter, John D., 2007. Matplotlib: A 2d graphics environment. Comput. Sci. Eng. 9 (3), 90–95.

Juschten, Maria, Ohnmacht, Timo, Thao, Vu Thi, Gerike, Regine, Hössinger, Reinhard, 2019. Carsharing in switzerland: identifying new markets by predicting membership based on data on supply and demand. Transportation 46 (4), 1171–1194.

Kaplan, Edward L., Meier, Paul, 1958. Nonparametric estimation from incomplete observations. J. Amer. Statist. Assoc. 53 (282), 457–481.

Katzman, Jared L., Shaham, Uri, Cloninger, Alexander, Bates, Jonathan, Jiang, Tingting, Kluger, Yuval, 2018. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. BMC Med. Res. Methodol. 18 (24).

Kostic, Bojan, Sourd, Romain Crastes dit, Hess, Stephane, Scheiner, Joachim, Holz-Rau, Christian, Pereira, Francisco Camara, 2020. Uncovering life-course patterns with causal discovery and survival analysis. arXiv e-prints, arXiv:2001.11399 [stat.AP].

Lee, M.-L.T., Whitmore, G.A., 2006. Threshold regression for survival analysis: Modeling event times by a stochastic process reaching a boundary. Statist. Sci. 21 (4), 501–513.

Lee, Changhee, Zame, William R., Yoon, Jinsung, Schaar, Mihaela van der, 2018. Deephit: A deep learning approach to survival analysis with competing risks. In: AAAI Conference on Artificial Intelligence (AAAI).

Li, Yexin, Zheng, Yu, 2019. Citywide bike usage prediction in a bike-sharing system. IEEE Trans. Knowl. Data Eng..

Lin, D.Y., 2007. On the breslow estimator. Lifetime Data Analysis 13, 471–480.

McKinney, Wes, Data structures for statistical computing in Python. In: Stéfan van der Walt and Jarrod Millman, editors, Proceedings of the 9th Python in Science Conference, 2010, pp. 51–56.

Müller, Johannes, Bogenberger, Klaus, 2015. Time series analysis of booking data of a free-floating carsharing system in berlin. Transp. Res. Proc. 10, 345–354.

2017. Openstreetmap contributors. Planet dump retrieved from https://planet.osm.org. Online; accessed June 28, 2019.

Pan, Yan, Zheng, Ray Chen, Zhang, Jiaxi, Yao, Xin, 2019. Predicting bike sharing demand using recurrent neural networks. Procedia Comput. Sci. 147, 562–566.

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

Pereira, Francisco Camara, Borysov, Stanislav S., 2018. Machine learning fundamentals. In: Mobility Patterns, Big Data and Transport Analytics. Elsevier, Amsterdam, The Netherlands, pp. 9–29, chapter 2.

Qian, Jia, Pianura, Livio, Comin, Matteo, 2018. Data-driven smart bike-sharing system by implementing machine learning algorithms. In: 2018 Sixth International Conference on Enterprise Systems (ES). IEEE, pp. 50–55.

Rossum, Guido van, Drake, Fred L., 2011. The Python Language Reference Manual. Network Theory Ltd., Boston, MA.

Ruffieux, Simon, Spycher, Nicolas, Mugellini, Elena, Khaled, Omar Abou, 2017. Real-time usage forecasting for bike-sharing systems: A study on random forest and convolutional neural network applicability. In: 2017 Intelligent Systems Conference (IntelliSys). IEEE, pp. 622–631.

Rumelhart, David E., Hinton, Geoffrey E., Williams, Ronald J., 1986a. Learning internal representations by error propagation. In: Rumelhart, David E., McClelland, James L., The P.D.P. Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations. The MIT Press, Cambridge, MA, pp. 318–362.

Rumelhart, David E., Hinton, Geoffrey E., Williams, Ronald J., 1986b. Learning representations by back-propagating errors. Nature 323, 533–536.

Schmöller, Stefan, Weikl, Simone, Müller, Johannes, Bogenberger, Klaus, 2015. Empirical analysis of free-floating carsharing usage: The munich and berlin case. Transp. Res. C 56, 34–51.

2020. Share NOW. https://www.share-now.com/. Online accessed 03 2020.

Sklar, Abe, 1973. Random variables, joint distribution functions and copulas. Kybernetika 9, 449–460.

Soares Machado, Cláudia A., Hue, Nicolas Patrick Marie de Salles, Berssaneti, Fernando Tobal, Quintanilha, José Alberto, 2018. An overview of shared mobility. Sustainability 10 (4342).

Sohi, Shahrom Hosseini, 2018. Data-Driven Time Series Demand Forecasting of Car-Sharing Services: The Case Study of DriveNow in Munich, Germany (Msc thesis). Technical University of Denmark/Technical University of Munich.

Tang, L., Thomas, L.C., Thomas, S., Bozzetto, J.-F., 2007. It's the economy stupid: modelling financial product purchases. Int. J. Bank Market. 25 (1), 22–38.

Technical University of Denmark, 2019. DTU climate station data. Online; accessed June 28, 2019.

Virtanen, Pauli, Gommers, Ralf, Oliphant, Travis E., Haberland, Matt, Reddy, Tyler, Cournapeau, David, Burovski, Evgeni, Peterson, Pearu, Weckesser, Warren, Bright, Jonathan, van der Walt, Stéfan J., Brett, Matthew, Wilson, Joshua, Jarrod Millman, K., Mayorov, Nikolay, Nelson, Andrew R.J., Jones, Eric, Kern, Robert, Larson, Eric, Carey, C.J., Polat, İlhan, Feng, Yu, Moore, Eric W., erPlas, Jake Vand, Laxalde, Denis, Perktold, Josef, Cimrman, Robert, Henriksen, Ian, Quintero, E.A., Harris, Charles R., Archibald, Anne M., Ribeiro, Antônio H., Pedregosa, Fabian, Mulbregt, Paul van, and SciPy 1.0 Contributors, 2020. Scipy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods 17, 261–272.

Vosooghi, Reza, Puchinger, Jakob, Jankovic, Marija, Sirin, Göknur, 2017. A critical analysis of travel demand estimation for new one-way carsharing systems. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 199–205.

Walt, Stéfan van der, Chris Colbert, S., Varoquaux, Gaël, 2011. The numpy array: A structure for efficient numerical computation. Comput. Sci. Eng. 13 (2), 22–30.

Wang, Bo, Kim, Inhi, 2018. Short-term prediction for bike-sharing service using machine learning. Transp. Res. Proc. 34, 171–178.

Wang, Ling, Liu, Qi, Ma, Wanjing, 2019. Optimization of dynamic relocation operations for one-way electric carsharing systems. Transp. Res. C 101, 55–69.