Real-time Demand Forecasting for an Urban Delivery Platform

Alexander Hess[a,1], Stefan Spinler[a,1], Matthias Winkenbach[b,1]

[a] *WHU - Otto Beisheim School of Management, Burgplatz 2, 56179 Vallendar, Germany*
[b] *Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, United States*

Abstract

Meal delivery platforms like Uber Eats shape the landscape in cities around the world. This paper addresses forecasting demand into the short-term future. We propose an approach incorporating both classical forecasting and machine learning methods. Model evaluation and selection is adapted to demand typical for such a platform (i.e., intermittent with a double-seasonal pattern). The results of an empirical study with a European meal delivery service show that machine learning models become competitive once the average daily demand passes a threshold. As a main contribution, the paper explains how a forecasting system must be set up to enable predictive routing.

Keywords: demand forecasting, intermittent demand, machine learning, urban delivery platform

[1]Emails: alexander.hess@whu.edu, stefan.spinler@whu.edu, mwinkenb@mit.edu

1. Introduction

In recent years, many meal delivery platform providers (e.g., Uber Eats, GrubHub, DoorDash, Deliveroo) with different kinds of business models have entered the markets in cities around the world. A study by [24] estimates the global market size to surpass 20 billion Dollars by 2025. A common feature of these platforms is that they do not operate kitchens but focus on marketing their partner restaurants' meals, unifying all order related processes in simple smartphone apps, and managing the delivery via a fleet of either employees or crowd-sourced sub-contractors.

Various kind of urban delivery platforms (UDP) have received attention in recent scholarly publications. [27] look into heuristics to simultaneously optimize courier scheduling and routing in general, while [35] do so for the popular dial-a-ride problem and [47] investigate the effect of different fulfillment strategies in the context of urban meal delivery. [19] and [1] focus their research on the routing aspect, which is commonly modeled as a so-called vehicle routing problem (VRP).

Not covered in the recent literature is research focusing on the demand forecasting problem a UDP faces. Due to the customers' fragmented locations and the majority of the orders occurring ad-hoc for immediate delivery in the case of a meal delivery platform, forecasting demand for the near future (i.e., several hours) and distinct locations of the city in real-time is an essential factor in achieving timely fulfillment. In general, demand forecasting is a well-researched discipline with a decades-long history in scholarly journals as summarized, for example, by [17]. Even some meal delivery platforms themselves publish their practices: For example, [4] provide a general overview of supply and demand forecasting at Uber and benchmarks of the methods used while [34] investigate how extreme events can be incorporated.

The conditions such platforms face are not limited to meal delivery: Any entity that performs ad-hoc requested point-to-point transportation at scale in an urban area benefits from a robust forecasting system. Examples include ride-hailing, such as the original Uber offering, or bicycle courier services. The common characteristics are:

- **Geospatial Slicing**: Forecasts for distinct parts of a city in parallel

- **Temporal Slicing**: Forecasts on a sub-daily basis (e.g., 60-minute windows)
- **Order Sparsity**: The historical order time series exhibit an intermittent pattern
- **Double Seasonality**: Demand varies with the day of the week and the time of day

Whereas the first two points can be assumed to vary with the concrete application's requirements, it is the last two that pose challenges for forecasting a platform's demand: Intermittent demand (i.e., many observations in the historic order time series exhibit no demand at all) renders most of the commonly applied error metrics useless. Moreover, many of the established forecasting methods can only handle a single and often low seasonality (i.e., repeated regular pattern), if at all.

In this paper, we develop a rigorous methodology as to how to build and evaluate a robust forecasting system for an urban delivery platform (UDP) that offers ad-hoc point-to-point transportation of any kind. We implement such a system with a broad set of commonly used forecasting methods. We not only apply established (i.e., "classical") time series methods but also machine learning (ML) models that have gained traction in recent years due to advancements in computing power and availability of larger amounts of data. In that regard, the classical methods serve as benchmarks for the ML methods. Our system is trained on and evaluated with a dataset obtained from an undisclosed industry partner that, during the timeframe of our study, was active in several European countries and, in particular, in France. Its primary business strategy is the delivery of meals from upper-class restaurants to customers in their home or work places via bicycles. In this empirical study, we identify the best-performing methods. Thus, we answer the following research questions:

Q1: Which forecasting methods work best under what circumstances?
Q2: How do classical forecasting methods compare with ML models?
Q3: How does the forecast accuracy change with more historic data available?
Q4: Can real-time information on demand be exploited?
Q5: Can external data (e.g., weather data) improve the forecast accuracy?

To the best of our knowledge, no such study has yet been published in a scholarly journal.

The subsequent Section 2 reviews the literature on the forecasting methods included in the system. Section 3 introduces our forecasting system, and Section 4 discusses the results obtained in the empirical study. Lastly, Section 5 summarizes our findings and concludes with an outlook on further research opportunities.

2. Literature Review

In this section, we review the specific forecasting methods that make up our forecasting system. We group them into classical statistics and ML models. The two groups differ mainly in how they represent the input data and how accuracy is evaluated.

A time series is a finite and ordered sequence of equally spaced observations. Thus, time is regarded as discrete and a time step as a short period. Formally, a time series Y is defined as $Y = \{y_t : t \in I\}$, or y_t for short, where I is an index set of positive integers. Besides its length $T = |Y|$, another property is the a priori fixed and non-negative periodicity k of a seasonal pattern in demand: k is the number of time steps after which a pattern repeats itself (e.g., $k = 12$ for monthly sales data).

2.1. Demand Forecasting with Classical Forecasting Methods

Forecasting became a formal discipline starting in the 1950s and has its origins in the broader field of statistics. [28] provide a thorough overview of the concepts and methods established, and [38] indicate business-related applications such as demand forecasting. These "classical" forecasting methods share the characteristic that they are trained over the entire Y first. Then, for prediction, the forecaster specifies the number of time steps for which he wants to generate forecasts. That is different for ML models.

2.1.1. Naïve Methods, Moving Averages, and Exponential Smoothing.

Simple forecasting methods are often employed as a benchmark for more sophisticated ones. The so-called naïve and seasonal naïve methods forecast the next time step in a time series, y_{T+1}, with the last observation, y_T, and, if a seasonal pattern is present, with the observation k steps before, y_{T+1-k}. As variants, both methods can be generalized to include drift terms in the presence of a trend or changing seasonal amplitude.

If a time series exhibits no trend, a simple moving average (SMA) is a generalization of the naïve method that is more robust to outliers. It is defined as follows: $\hat{y}_{T+1} = \frac{1}{h} \sum_{i=T-h}^{T} y_i$

where h is the horizon over which the average is calculated. If a time series exhibits a seasonal pattern, setting h to a multiple of the periodicity k suffices that the forecast is unbiased.

Starting in the 1950s, another popular family of forecasting methods, so-called exponential smoothing methods, was introduced by [12], [26], and [48]. The idea is that forecasts $\hat{y}_{T+1}$ are a weighted average of past observations where the weights decay over time; in the case of the simple exponential smoothing (SES) method we obtain: $\hat{y}_{T+1} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots + \alpha(1-\alpha)^{T-1}y_1$ where α (with $0 \leq \alpha \leq 1$) is a smoothing parameter.

Exponential smoothing methods are often expressed in an alternative component form that consists of a forecast equation and one or more smoothing equations for unobservable components. Below, we present a generalization of SES, the so-called Holt-Winters' seasonal method, in an additive formulation. ℓ_t, b_t, and s_t represent the unobservable level, trend, and seasonal components inherent in y_t, and β and γ complement α as smoothing parameters:

$$\begin{aligned}
\hat{y}_{t+1} &= \ell_t + b_t + s_{t+1-k} \\
\ell_t &= \alpha(y_t - s_{t-k}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-k}
\end{aligned}$$

With b_t, s_t, β, and γ removed, this formulation reduces to SES. Distinct variations exist: Besides the three components, [20] add dampening for the trend, [39] provides multiplicative formulations, and [43] adds dampening to the latter. The accuracy measure commonly employed is the sum of squared errors between the observations and their forecasts.

Originally introduced by [2], [29] show how the Theta method can be regarded as an equivalent to SES with a drift term. We mention this method here only because [4] emphasize that it performs well at Uber. However, in our empirical study, we find that this is not true in general.

[32] introduce statistical processes, so-called innovations state-space models, to generalize the methods in this sub-section. They call this family of models ETS as they capture error,

trend, and seasonal terms. Linear and additive ETS models have a structure like so:

$$y_t = \vec{w} \cdot \vec{x}_{t-1} + \epsilon_t$$
$$\vec{x_t} = \boldsymbol{F}\vec{x}_{t-1} + \vec{g}\epsilon_t$$

y_t denote the observations as before while $\vec{x}_t$ is a state vector of unobserved components. ϵ_t is a white noise series and the matrix $\boldsymbol{F}$ and the vectors $\vec{g}$ and $\vec{w}$ contain a model's coefficients. Just as the models in the next sub-section, ETS models are commonly fitted with maximum likelihood and evaluated using information theoretical criteria against historical data. We refer to [31] for a thorough summary.

2.1.2. Autoregressive Integrated Moving Averages.

[6], [7], and more papers by the same authors in the 1960s introduce a type of model where observations correlate with their neighbors and refer to them as autoregressive integrated moving average (ARIMA) models for stationary time series. For a thorough overview, we refer to [8] and [11].

A time series y_t is stationary if its moments are independent of the point in time where it is observed. A typical example is a white noise ϵ_t series. Therefore, a trend or seasonality implies non-stationarity. [33] provide a test to check the null hypothesis of stationary data. To obtain a stationary time series, one chooses from several techniques: First, to stabilize a changing variance (i.e., heteroscedasticity), one applies a Box-Cox transformation (e.g., log) as first suggested by [5]. Second, to factor out a trend (or seasonal) pattern, one computes differences of consecutive (or of lag k) observations or even differences thereof. Third, it is also common to pre-process y_t with one of the decomposition methods mentioned in Sub-section 2.1.3 below with an ARIMA model then trained on an adjusted y_t.

In the autoregressive part, observations are modeled as linear combinations of its predecessors. Formally, an $AR(p)$ model is defined with a drift term c, coefficients ϕ_i to be estimated (where i is an index with $0 < i \leq p$), and white noise ϵ_t like so: $AR(p): \quad y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t$. The moving average part considers observations to be regressing towards a linear combination of past forecasting errors. Formally, a $MA(q)$ model is defined with a drift term c, coefficients θ_j to be estimated, and white noise terms ϵ_t (where j is an index with $0 < j \leq q$) as follows: $MA(q): \quad y_t =$

$c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$. Finally, an $ARIMA(p, d, q)$ model unifies both parts and adds differencing where d is the degree of differences and the $'$ indicates differenced values: $ARIMA(p, d, q): \quad y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$.

$ARIMA(p, d, q)$ models are commonly fitted with maximum likelihood estimation. To find an optimal combination of the parameters p, d, and q, the literature suggests calculating an information theoretical criterion (e.g., Akaike's Information Criterion) that evaluates the fit on historical data. [30] provide a step-wise heuristic to choose p, d, and q, that also decides if a Box-Cox transformation is to be applied, and if so, which one. To obtain a one-step-ahead forecast, the above equation is reordered such that t is substituted with $T + 1$. For forecasts further into the future, the actual observations are subsequently replaced by their forecasts. Seasonal ARIMA variants exist; however, the high frequency k in the kind of demand a UDP faces typically renders them impractical as too many coefficients must be estimated.

2.1.3. Seasonal and Trend Decomposition using Loess.

A time series y_t may exhibit different types of patterns; to fully capture each of them, the series must be decomposed. Then, each component is forecast with a distinct model. Most commonly, the components are the trend t_t, seasonality s_t, and remainder r_t. They are themselves time series, where only s_t exhibits a periodicity k. A decomposition may be additive (i.e., $y_t = s_t + t_t + r_t$) or multiplicative (i.e., $y_t = s_t * t_t * r_t$); the former assumes that the effect of the seasonal component is independent of the overall level of y_t and vice versa. The seasonal component is centered around 0 in both cases such that its removal does not affect the level of y_t. Often, it is sufficient to only seasonally adjust the time series, and model the trend and remainder together, for example, as $a_t = y_t - s_t$ in the additive case.

Early approaches employed moving averages (cf., Sub-section 2.1.1) to calculate a trend component, and, after removing that from y_t, averaged all observations of the same seasonal lag to obtain the seasonal component. The downsides of this are the subjectivity in choosing the window lengths for the moving average and the seasonal averaging, the incapability of the seasonal component to vary its amplitude over time, and the missing handling of outliers.

The X11 method developed at the U.S. Census Bureau and described in detail by [16] overcomes these disadvantages. However, due to its background in economics, it is designed primarily for quarterly or monthly data, and the change in amplitude over time cannot be controlled. Variants of this method are the SEATS decomposition by the Bank of Spain and the newer X13-SEATS-ARIMA method by the U.S. Census Bureau. Their main advantages stem from the fact that the models calibrate themselves according to statistical criteria without manual work for a statistician and that the fitting process is robust to outliers.

[15] introduce a seasonal and trend decomposition using a repeated locally weighted regression - the so-called Loess procedure - to smoothen the trend and seasonal components, which can be viewed as a generalization of the methods above and is denoted by the acronym STL. In contrast to the X11, X13, and SEATS methods, the STL supports seasonalities of any lag k that must, however, be determined with additional statistical tests or set with out-of-band knowledge by the forecaster (e.g., hourly demand data implies $k = 24 * 7 = 168$ assuming customer behavior differs on each day of the week). Moreover, the seasonal component's rate of change, represented by the ns parameter and explained in detail with Figure 3 in Section 3, must be set by the forecaster as well, while the trend's smoothness may be controlled via setting a non-default window size. Outliers are handled by assignment to the remainder such that they do not affect the trend and seasonal components. In particular, the manual input needed to calibrate the STL explains why only the X11, X13, and SEATS methods are widely used by practitioners. However, the widespread adoption of concepts like cross-validation (cf., Sub-section 2.2.2) in recent years enables the usage of an automated grid search to optimize the parameters. The STL's usage within a grid search is facilitated even further by its being computationally cheaper than the other methods discussed.

2.2. Demand Forecasting with Machine Learning Methods

ML methods have been employed in all kinds of prediction tasks in recent years. In this section, we restrict ourselves to the models that performed well in our study: Random Forest (RF) and Support Vector Regression (SVR). RFs are in general well-suited for datasets without a priori knowledge about the patterns, while SVR is known to perform well on time series data, as shown by [21] in general and [3] specifically for intermittent demand. Gradient Boosting, another popular ML method, was consistently outperformed by RFs, and artificial neural networks require an amount of data exceeding what our industry partner has by far.

2.2.1. Supervised Learning.

A conceptual difference between classical and ML methods is the format for the model inputs. In ML models, a time series Y is interpreted as labeled data. Labels are collected into a vector $\vec{y}$ while the corresponding predictors are aligned in an $(T-n)\times n$ matrix $\boldsymbol{X}$:

$$\vec{y} = \begin{pmatrix} y_T \\ y_{T-1} \\ \cdots \\ y_{n+1} \end{pmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} y_{T-1} & y_{T-2} & \cdots & y_{T-n} \\ y_{T-2} & y_{T-3} & \cdots & y_{T-(n+1)} \\ \cdots & \cdots & \cdots & \cdots \\ y_n & y_{n-1} & \cdots & y_1 \end{bmatrix}$$

The $m = T - n$ rows are referred to as samples and the n columns as features. Each row in $\boldsymbol{X}$ is "labeled" by the corresponding entry in $\vec{y}$, and ML models are trained to fit the rows to their labels. Conceptually, we model a functional relationship f between $\boldsymbol{X}$ and $\vec{y}$ such that the difference between the predicted $\vec{\hat{y}} = f(\boldsymbol{X})$ and the true $\vec{y}$ are minimized according to some error measure $L(\vec{\hat{y}}, \vec{y})$, where L summarizes the goodness of the fit into a scalar value (e.g., the well-known mean squared error [MSE]; cf., Section 3). $\boldsymbol{X}$ and $\vec{y}$ show the ordinal character of time series data: Not only overlap the entries of $\boldsymbol{X}$ and $\vec{y}$, but the rows of $\boldsymbol{X}$ are shifted versions of each other. That does not hold for ML applications in general (e.g., the classical example of predicting spam vs. no spam emails, where the features model properties of individual emails), and most of the common error measures presented in introductory texts on ML, are only applicable in cases without such a structure in $\boldsymbol{X}$ and $\vec{y}$. n, the number of past time steps required to predict a y_t, is an exogenous model parameter. For prediction, the forecaster supplies the trained ML model an input vector in the same format as a row $\vec{x}_i$ in $\boldsymbol{X}$. For example, to predict y_{T+1}, the model takes the vector $(y_T, y_{T-1}, ..., y_{T-n+1})$ as input. That is in contrast to the classical methods, where we only supply the number of time steps to be predicted as a scalar integer.

2.2.2. Cross-Validation.

Because ML models are trained by minimizing a loss function L, the resulting value of L underestimates the true error we see when predicting into the actual future by design. To counter that, one popular and model-agnostic approach is cross-validation (CV), as summarized, for example, by [22]. CV is a resampling technique, which ranomdly splits the samples into a training and a test set. Trained on the former, an ML model makes forecasts on the latter. Then, the value of L calculated only on the test set gives a realistic and unbiased estimate of the true forecasting error, and may be used for one of two distinct aspects: First, it assesses the quality of a fit and provides an idea as to how the model would perform in production when predicting into the actual future. Second, the errors of models of either different methods or the same method with different parameters may be compared with each other to select the best model. In order to first select the best model and then assess its quality, one must apply two chained CVs: The samples are divided into training, validation, and test sets, and all models are trained on the training set and compared on the validation set. Then, the winner is retrained on the union of the training and validation sets and assessed on the test set.

Regarding the splitting, there are various approaches, and we choose the so-called k-fold CV, where the samples are randomly divided into k folds of the same size. Each fold is used as a test set once and the remaining $k-1$ folds become the corresponding training set. The resulting k error measures are averaged. A k-fold CV with $k=5$ or $k=10$ is a compromise between the two extreme cases of having only one split and the so-called leave-one-out CV where $k=m$: Computation is still relatively fast and each sample is part of several training sets maximizing the learning from the data. We adapt the k-fold CV to the ordinal stucture in $\boldsymbol{X}$ and $\vec{y}$ in Sub-section 3.

2.2.3. Random Forest Regression.

[10] introduce the classification and regression tree (CART) model that is built around the idea that a single binary decision tree maps learned combinations of intervals of the feature columns to a label. Thus, each sample in the training set is associated with one leaf node that is reached by following the tree from its root and branching along the arcs according to some learned splitting rule per intermediate node that compares the sample's realization for the feature specified by the rule to the learned decision rule. While such models are computationally fast and offer a high degree of interpretability, they tend to overfit strongly to the training set as the splitting rules are not limited to any functional form (e.g., linear) in the relationship between the features and the labels. In the regression case, it is common

to maximize the variance reduction I_V from a parent node N to its two children, $C1$ and $C2$, as the splitting rule. [10] formulate this as follows:

$$I_V(N) = \frac{1}{|S_N|^2} \sum_{i \in S_N} \sum_{j \in S_N} \frac{1}{2}(y_i - y_j)^2 - \left(\frac{1}{|S_{C1}|^2} \sum_{i \in S_{C1}} \sum_{j \in S_{C1}} \frac{1}{2}(y_i - y_j)^2 + \frac{1}{|S_{C2}|^2} \sum_{i \in S_{C2}} \sum_{j \in S_{C2}} \frac{1}{2}(y_i - y_j)^2 \right)$$

S_N, S_{C1}, and S_{C2} are the index sets of the samples in N, $C1$, and $C2$.

[25] and then [9] generalize this method by combining many CART models into one forest of trees where every single tree is a randomized variant of the others. Randomization is achieved at two steps in the training process: First, each tree receives a distinct training set resampled with replacement from the original training set, an idea also called bootstrap aggregation. Second, at each node a random subset of the features is used to grow the tree. Trees can be fitted in parallel speeding up the training significantly. For prediction at the tree level, the average of all the samples at a particular leaf node is used. Then, the individual values are combined into one value by averaging again across the trees. Due to the randomization, the trees are decorrelated offsetting the overfitting. Another measure to counter overfitting is pruning the tree, either by specifying the maximum depth of a tree or the minimum number of samples at leaf nodes.

The forecaster must tune the structure of the forest. Parameters include the number of trees in the forest, the size of the random subset of features, and the pruning criteria. The parameters are optimized via grid search: We train many models with parameters chosen from a pre-defined list of values and select the best one by CV. RFs are a convenient ML method for any dataset as decision trees do not make any assumptions about the relationship between features and labels. [23] use RFs to predict the hourly demand for water in an urban context, a similar application as the one in this paper, and find that RFs work well with time series type of data.

2.2.4. Support Vector Regression.

[46] and [45] introduce the so-called support vector machine (SVM) model, and [44] summarizes the research conducted since then. In its basic version, SVMs are linear classifiers, modeling a binary decision, that fit a hyperplane into the feature space of $\boldsymbol{X}$ to maximize the margin around the hyperplane seperating the two groups of labels. SVMs were popularized in the 1990s in the context of optical character recognition, as shown in [40].

[18] and [42] adapt SVMs to the regression case, and [41] provide a comprehensive introduction thereof. [36] and [37] focus on SVRs in the context of time series data and find that they tend to outperform classical methods. [13] and [14] apply SVRs to predict the hourly demand for water in cities, an application similar to the UDP case.

In the SVR case, a linear function $\hat{y}_i = f(\vec{x}_i) = \langle \vec{w}, \vec{x}_i \rangle + b$ is fitted so that the actual labels y_i have a deviation of at most ϵ from their predictions $\hat{y}_i$ (cf., the constraints below). SVRs are commonly formulated as quadratic optimization problems as follows:

$$\text{minimize } \frac{1}{2}\left\|\vec{w}\right\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \quad \text{subject to } \begin{cases} y_i - \langle \vec{w}, \vec{x}_i \rangle - b \leq \epsilon + \xi_i, \\ \langle \vec{w}, \vec{x}_i \rangle + b - y_i \leq \epsilon + \xi_i^* \end{cases}$$

$\vec{w}$ are the fitted weights in the row space of $\boldsymbol{X}$, b is a bias term in the column space of $\boldsymbol{X}$, and $\langle \cdot, \cdot \rangle$ denotes the dot product. By minimizing the norm of $\vec{w}$, the fitted function is flat and not prone to overfitting strongly. To allow individual samples outside the otherwise hard ϵ bounds, non-negative slack variables ξ_i and ξ_i^* are included. A non-negative parameter C regulates how many samples may violate the ϵ bounds and by how much. To model non-linear relationships, one could use a mapping $\Phi(\cdot)$ for the $\vec{x}_i$ from the row space of $\boldsymbol{X}$ to some higher dimensional space; however, as the optimization problem only depends on the dot product $\langle \cdot, \cdot \rangle$ and not the actual entries of $\vec{x}_i$, it suffices to use a kernel function k such that $k(\vec{x}_i, \vec{x}_j) = \langle \Phi(\vec{x}_i), \Phi(\vec{x}_j) \rangle$. Such kernels must fulfill certain mathematical properties, and, besides polynomial kernels, radial basis functions with $k(\vec{x}_i, \vec{x}_j) = exp(\gamma \left\|\vec{x}_i - \vec{x}_j\right\|^2)$ are a popular candidate where γ is a parameter controlling for how the distances between any two samples influence the final model. SVRs work well with sparse data in high dimensional spaces, such as intermittent demand data, as they minimize the risk of misclassification or predicting a significantly far off value by maximizing the error margin, as also noted by [3].

3. Model Formulation

4. Empirical Study: A Meal Delivery Platform in Europe

5. Conclusion

Glossary

CART Classification and Regression Trees. 10

CV Cross Validation. 9

ML Machine Learning. 3

RF Random Forest. 8

STL Seasonal and Trend Decomposition using Loess. 8

SVM Support Vector Machine. 11

SVR Support Vector Regression. 8

UDP Urban Delivery Platform. 3

VRP Vehicle Routing Problem. 2

References

[1] Alcaraz, J.J., Caballero-Arnaldos, L., Vales-Alonso, J., 2019. Rich vehicle routing problem with last-mile outsourcing decisions. Transportation Research Part E: Logistics and Transportation Review 129, 263–286.

[2] Assimakopoulos, V., Nikolopoulos, K., 2000. The theta model: a decomposition approach to forecasting. International Journal of Forecasting 16, 521–530.

[3] Bao, Y., Wang, W., Zhang, J., 2004. Forecasting intermittent demand by svms regression, in: 2004 IEEE International Conference on Systems, Man and Cybernetics, pp. 461–466.

[4] Bell, F., Smyl, S., 2018. Forecasting at uber: An introduction. `https://eng.uber.com/forecasting-introduction/`. Accessed: 2020-10-01.

[5] Box, G., Cox, D., 1964. An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological) 26, 211–252.

[6] Box, G., Jenkins, G., 1962. Some statistical aspects of adaptive optimization and control. Journal of the Royal Statistical Society. Series B (Methodological) 24, 297–343.

[7] Box, G., Jenkins, G., 1968. Some recent advances in forecasting and control. Journal of the Royal Statistical Society. Series C (Applied Statistics) 17, 91–109.

[8] Box, G., Jenkins, G., Reinsel, G., Ljung, G., 2015. Time Series Analysis: Forecasting and Control. Wiley Series in Probability and Statistics, Wiley.

[9] Breiman, L., 2001. Random forests. Machine Learning 45, 5–32.

[10] Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth.

[11] Brockwell, P., Davis, R., 2016. Introduction to Time Series and Forecasting. Springer Texts in Statistics, Springer.

[12] Brown, R., 1959. Statistical Forecasting for Inventory Control. McGraw/Hill.

[13] Chen, L., Zhang, T.q., 2006a. Hourly water demand forecast model based on bayesian least squares support vector machine. Journal of Tianjin University 39, 1037–1042.

[14] Chen, L., Zhang, T.q., 2006b. Hourly water demand forecast model based on least squares support vector machine. Journal of Harbin Institute of Technology 38, 1528–1530.

[15] Cleveland, R., Cleveland, W., McRae, J., Terpenning, I., 1990. Stl: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics 6, 3–73.

[16] Dagum, E., Bianconcini, S., 2016. Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation. Statistics for Social and Behavioral Sciences, Springer.

[17] De Gooijer, J., Hyndman, R., 2006. 25 years of time series forecasting. International Journal of Forecasting 22, 443–473.

[18] Drucker, H., Burges, C., Kaufman, L., Smola, A., Vapnik, V., 1997. Support vector regression machines, in: Advances in Neural Information Processing Systems, Springer. pp. 155–161.

[19] Ehmke, J.F., Campbell, A.M., Thomas, B.W., 2018. Optimizing for total costs in vehicle routing in urban areas. Transportation Research Part E: Logistics and Transportation Review 116, 242–265.

[20] Gardner, E., McKenzie, E., 1985. Forecasting trends in time series. Management Science 31, 1237–1246.

[21] Hansen, J., McDonald, J., Nelson, R., 2006. Some evidence on forecasting time-series with support vector machines. Journal of the Operational Research Society 57, 1053–1063.

[22] Hastie, T., Tibshirani, R., Friedman, J., 2013. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[23] Herrera, M., Torgo, L., Izquierdo, J., Pérez-García, R., 2010. Predictive models for forecasting hourly urban water demand. Journal of Hydrology 387, 141–150.

[24] Hirschberg, C., Rajko, A., Schumacher, T., Wrulich, M., 2016. Mckinsey: The changing market for food delivery. https://www.mckinsey.com/industries/high-tech/our-insights/the-changing-market-for-food-delivery. Accessed: 2020-10-01.

[25] Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 832–844.

[26] Holt, C., 1957. Forecasting seasonals and trends by exponentially weighted moving averages. ONR Memorandum 52.

[27] Hou, L., Li, D., Zhang, D., 2018. Ride-matching and routing optimisation: Models and a large neighbourhood search heuristic. Transportation Research Part E: Logistics and Transportation Review 118, 143–162.

[28] Hyndman, R., Athanasopoulos, G., 2018. Forecasting: Principles and Practice. OTexts.

[29] Hyndman, R., Billah, B., 2003. Unmasking the theta method. International Journal of Forecasting 19, 287–290.

[30] Hyndman, R., Khandakar, Y., 2008. Automatic time series forecasting: The forecast package for r. Journal of Statistical Software 26.

[31] Hyndman, R., Koehler, A., Ord, K., Snyder, R., 2008. Forecasting with Exponential Smoothing: the State Space Approach. Springer.

[32] Hyndman, R., Koehler, A., Snyder, R., Grose, S., 2002. A state space framework for automatic forecasting using exponential smoothing methods. International Journal of Forecasting 18, 439–454.

[33] Kwiatkowski, D., Phillips, P., Schmidt, P., Shin, Y., 1992. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? Journal of Econometrics 54, 159–178.

[34] Laptev, N., Smyl, S., Shanmugam, S., 2017. Engineering extreme event forecasting at uber with recurrent neural networks. https://eng.uber.com/neural-networks/. Accessed: 2020-10-01.

[35] Masmoudi, M.A., Hosny, M., Demir, E., Genikomsakis, K.N., Cheikhrouhou, N., 2018. The dial-a-ride problem with electric vehicles and battery swapping stations. Transportation research part E: logistics and transportation review 118, 392–420.

[36] Müller, K.R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V., 1997. Predicting time series with support vector machines, in: International Conference on Artificial Neural Networks, Springer. pp. 999–1004.

[37] Müller, K.R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V., 1999. Using support vector machines for time series prediction. Advances in Kernel Methods — Support Vector Learning , 243–254.

[38] Ord, K., Fildes, R., Kourentzes, N., 2017. Principles of Business Forecasting. WESSEX Press.

[39] Pegels, C., 1969. Exponential forecasting: Some new variations. Management Science 15, 311–315.

[40] Schölkopf, B., Knirsch, P., Smola, A., Burges, C., 1998. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces, in: Mustererkennung 1998. Springer, pp. 125–132.

[41] Smola, A., Schölkopf, B., 2004. A tutorial on support vector regression. Statistics and Computing 14, 199–222.

[42] Stitson, M., Gammerman, A., Vapnik, V., Vovk, V., Watkins, C., Weston, J., 1999. Support vector regression with anova decomposition kernels. Advances in Kernel Methods — Support Vector Learning , 285–292.

[43] Taylor, J., 2003. Exponential smoothing with a damped multiplicative trend. International Journal of Forecasting 19, 715–725.

[44] Vapnik, V., 2013. The Nature of Statistical Learning Theory. Springer.

[45] Vapnik, V., Chervonenkis, A., 1964. A note on one class of perceptrons. Automation and Remote Control 25.

[46] Vapnik, V., Lerner, A., 1963. Pattern recognition using generalized portrait method. Automation and Remote Control 24, 774–780.

[47] Wang, Z., 2018. Delivering meals for multiple suppliers: Exclusive or sharing logistics service. Transportation Research Part E: Logistics and Transportation Review 118, 496–512.

[48] Winters, P., 1960. Forecasting sales by exponentially weighted moving averages. Management Science 6, 324–342.