Developing a Streaming-based Architecture for Demand Prediction of Taxi Trips in the Presence of Concept Drift

Master Thesis

by

Marcel Hofmann

Matriculation number: 2069630

At the Department of Economics and Management

Karlsruhe Service Research Institute

Advisor:	Prof. Dr. Gerhard Satzger
Second Advisor:	Prof. Dr. Hansjörg Fromm
External Advisors:	Stefan Igel (inovex GmbH)
	Hans-Peter Zorn (inovex GmbH)
Date of Submission:	20.05.2019

Declaration of Academic Integrity

I hereby confirm that the present thesis is solely my own work and that if any text passages or diagrams from books, papers, the Web or other sources have been copied or in any other way used, all references—including those found in electronic media—have been acknowledged and fully cited.

Karlsruhe, 20.05.2019 *Marcel Hofmann*

Abstract

Machine learning models are omnipresent for predictions on data streams. One challenge of deployed models is the change of data over time – a phenomenon called concept drift. If not considered in the entire process, from model design to deployment, a concept drift can lead to significant mispredictions. In this thesis the effects of concept drift in regression tasks are explored. A novel approach for concept drift handling is introduced, which depicts a strategy to switch between the application of simple and complex machine learning models. The approach leverages the individual strengths of each model, switching to the simpler model if a sudden drift occurs and switching back to the complex model for typical situations. To evaluate the approach, it is instantiated on a real-world dataset of taxi demand in New York City. This dataset is prone to multiple drifts, e.g. the weather phenomena of blizzards, resulting in a sudden decrease of taxi demand, or festival taking place, which results in unusually high demand. The analysis of the approach over a time span of six years shows that the suggested approach outperforms all regarded baselines significantly.

Contents

Acronyms

ACF	Autocorrelation function
ADWIN	Adaptive sliding window
ARIMA	Autoregressive integrated moving average
BOA	Biased Error Overlap Approach
CRISP-DM	Cross industry standard process for data mining
DM-Test	Diebold-Mariano test
DDM	Dirft detection method
EDDM	Early dirft detection method
EIA	Error intersection approach
EWMA	Exponentially weighted moving average
LSTM	Long short term memory
ML	Machine learning
MLP	Multilayer perceptron
NNAR	Neural network autoregression model
NYC	New York City
PAC	Probably approximately correct
pACF	Partial autocorrelation function
PH	Page-Hinkley
sMAPE	Symmetric mean absolute percentage error
RMSE	Root mean squared error
RNN	Recurrent neural network
TLC	New York City Taxi & Limousine Commission

List of Figures

List of Tables

1 Introduction

In the last decade the amount of data available to businesses has continued to grow and reaping the benefits of this resource has become a necessity in all industries. Machine Learning (ML) is playing an important role in this context, helping to transform and (semi-)automate established business processes – spanning from marketing to operations (Chen et al., 2012). Typical applications of ML range from computer vision over speech recognition to natural language processing or the control of manufacturing robots. Thereby, these techniques are especially influencing data-intensive tasks such as consumer services or the analysis and handling of faults in complex production systems (Jordan and Mitchell, 2015).

ML can create ongoing value when the resulting models are deployed in the information systems of the respective company and deliver ongoing recommendations and optimized decisions on continuous data streams (Dunning and Friedman, 2017). However, data streams usually change over time and thus, their underlying probability distribution or their data structure changes (Wang and Abraham, 2015). For example, many data streams describing human behaviour are bound to shift over time, as influencing factors like trends and preferences change (Žliobaitė et al., 2016). The challenge of changing data streams for supervised ML tasks has been described with the term *concept drift* (Widmer and Kubat, 1996). Most research on concept drift has been focused on classification tasks, and the minority of published papers considers regression tasks (Cavalcante et al., 2016; Baier et al., 2019). Additionally, most of the work on concept drift uses artificial datasets to evaluate new approaches or is focused only on a small subset of publicly available datasets, like the Australian Electricity Market or the Airplane dataset (Gama et al., 2014). While this allows for objective comparisons, the emphasis on simulated problems means that many real-world challenges are ignored. "Real" data is noisy, contains errors, and it is often unclear what to look for (L'Heureux et al., 2017). Therefore, this work focuses on the application of concept drift strategies for regression tasks on an unexplored real-world dataset which leads to the first research question of this work:

RQ1: How can we address concept drift in regression problems in a real-world context?

For answering this question the application domain of demand forecasts is considered using the publicly available dataset of every taxi trip performed in New York City since 2009 (TLC, 2019). By predicting the number of taxis needed in certain regions of the city at a specific time, the waiting times can be reduced, increasing efficiency and customer satisfaction. With increasing competition in the mobility market (e.g. Uber, Lyft, Lime), being able to anticipate demand may be a decisive factor (Zhao et al., 2016). Given the long time span covered by the dataset and the many hidden variables that can influence taxi demand, different types of drift can be observed, which indicates its suitability for the task at hand. The first part of this research will focus on detecting and describing drifting concepts and evaluating existing approaches on this regression task.

An initial analysis of several predictive models shows that during certain short periods a simple baseline model was significantly more accurate than more sophisticated approaches. In their proposal for a machine learning model management architecture, called "Rendezvous-Architecture", Dunning and Friedman (2017) include a so-called "Canary Model". Named after the canary bird used by miners to detect harmful gases, this model is used to "detect shifts in the input data and as a comparison benchmark for other models" (Dunning and Friedman, 2017, p.38). Inspired by this approach of leveraging an older or simpler model, a second research question emerged:

RQ2: How can we leverage models with different degrees of complexity to make the prediction more robust?

In answering this question a new approach is proposed – named *error intersection approach* (EIA), which utilizes static prediction models which are alternated based on the development of the error curve. Static models have the advantage that they need to be implemented only once and can also be verified and tested extensively before they are deployed in production for ongoing predictions. This approach is evaluated against detectors for sudden drift, both on its accuracy overall, as well as the behaviour during specific time frames.

The thesis is organized as follows: Chapter 2 analyzes the current state of relevant research in the areas of concept drift, time series prediction and traffic demand

prediction. Chapter 3 describes the design and implementation of predictors and drift detectors that are used to answer the research questions. Chapter 4 evaluates the accuracy of these predictors and detectors over a time span of six years. Finally, a conclusion is drawn in Chapter 5 on the implications of this work and areas of future research are outlined.

2 Theoretical Foundations

In this chapter the most important concepts and definitions for the thesis are discussed. The aim is to build towards an understanding of the problem domain, the current state of research and its most recent developments.

2.1 Time Series Forecasting

Time series forecasting, or time series prediction, is used to make predictions over many dynamic processes, such as stock price movements, monthly sales numbers or temperature changes in a city (Cavalcante et al., 2016). While the research area of time series analysis and forecasting is still continuously evolving, this thesis will focus on equidistant time series:

Definition 1: Equidistant Time Series: *A time series is a sequence of observations, the order of which is based on time. Every value* y_t *of a time series* $Y = (y_t)_{t \in T} = (y_0, y_1, \ldots, y_T)$ *is assigned a point in time* $t \in \{0, \ldots, T\}$*. The series is equidistant, if the distance between consecutive points in time is constant.*

Figure 1 shows a well-known equidistant time series of monthly totals of international airline passengers from 1949 to 1960, from Box et al. (1970). It will be used throughout this chapter as an example to visualize the presented concepts.

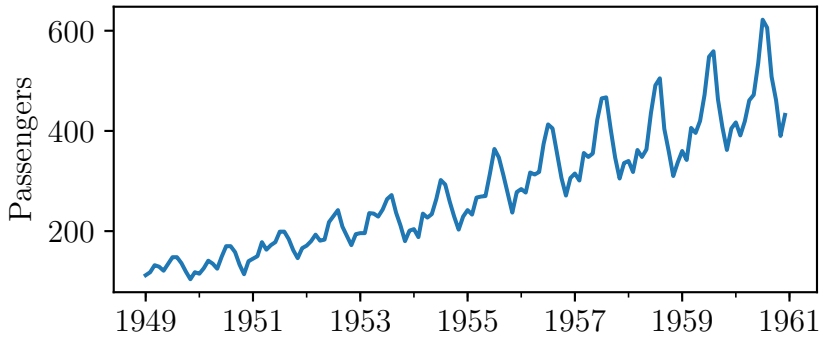

Figure 1: Plot of the airline passengers dataset

Time series forecasting can then be defined as the task of extrapolating future realisations of a given time series over a *prediction window*. Given the fact, that a time

series is time-dependent, a naïve approach is to simply assume the next value $\hat{y}$ (or values) to be equal to the last known one:

$$\hat{y}_{T+1} = y_T \tag{2.1}$$

This basic idea is commonly extended to include known, simple trends, such as inflation. Though not very sophisticated (and not good in its prediction), this model is commonly used as a baseline, to help evaluate more complex models (Hyndman and Athanasopoulos, 2013).

The naïve approach is based on the assumption, that a time series is autocorrelated. Autocorrelation is the correlation between observations of a certain distance. The autocorrelation coefficient r_k is defined as:

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T} (y_t - \overline{y})^2} \tag{2.2}$$

where $\bar{y}$ is the mean of all observations. Calculating the coefficient for many lags k yields a correlation plot as shown in Figure 2 for the airline passenger dataset. Each point represents a autocorrelation coefficient r_k. A point that falls outside the blue band is statistically significantly different from 0 with $\alpha = 0.05$. The overall shape of this particular plot can be considered typical for seasonal data with a trend. The seasonality causes the up-and-down; the trend causes the coefficient to decrease with larger lags (Hyndman and Athanasopoulos, 2013).

Another important concept of time series forecasting is stationarity. A time series is stationary, if the value of the time series does not depend on time:

Definition 2: (Weakly) Stationary time series: *A time series* $(y_t)_{t\in\mathrm{T}}$ *is called (weakly) stationary if*

i) the expected value is constant, i.e. $E(y_t) = \mu$ *for all* $t \in \mathbb{T}$

ii) the variance is finite, i.e. $\mathrm{Var}(y_t) < \infty$ *for all* $t \in \mathbb{T}$*, and*

iii) the autocovariance is stable against shifts, i.e. $\mathrm{Cov}(y_{t_1}, y_{t_2}) = \mathrm{Cov}(y_{s+t_1}, y_{s+t_2})$ *for all* $s, t_1, t_2 \in \mathbb{T}$

When analyzing a stationary time series, properties can be found that do not only apply to certain observations, but instead are invariant to time. Transforming a

time series into a stationary one is often a first step, since many models assume a stationary series. One example, the ARIMA model, will be explained in more detail in Section 2.1.3.

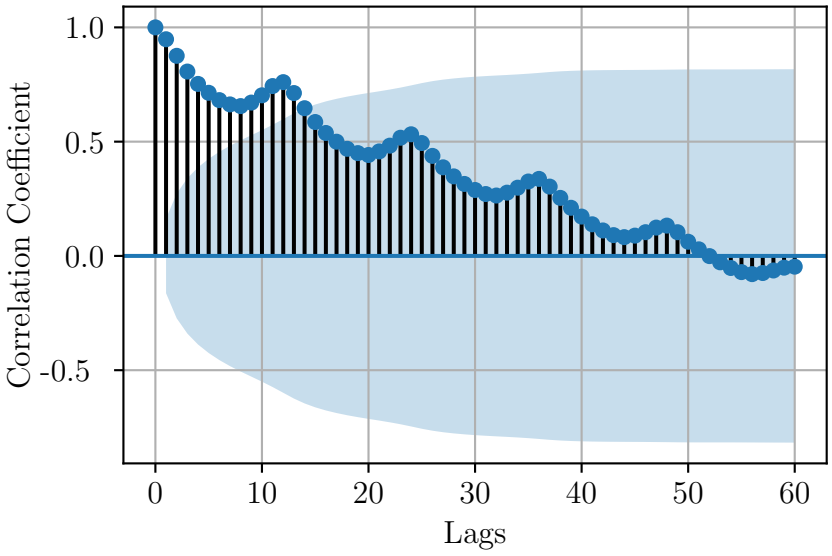

Figure 2: Autocorrelation plot of the airline passengers dataset

2.1.1 Classical Decomposition

For most economic use-cases it can be assumed that the observation follows an overall trend as well as some seasonality, or regularly recurring pattern. This can be modelled as a set of overlaying components (Brockwell and Davis, 2002):

- **Trend component** T_t: shows the long-term trend
- **Seasonal component** S_t: commonly has a wave shape and describes recurring developments
- **Random noise component** u_t: is the result of irregularly changing other influences and often described as *white noise*, with a mean of zero and constant variance.

Based on these components the family of decomposition models was developed. Decomposition by itself is only a framework, describing how to separate the series into its components. However, functions and assumptions of each component can be

chosen freely. There are two different types of time series decomposition - additive (often called *linear*) and multiplicative (often called *nonlinear*):

$$\begin{aligned} y_t &= T_t + S_t + u_t \quad \text{additive model} \\ y_t &= T_t \times S_t \times u_t \quad \text{multiplicative model} \end{aligned} \tag{2.3}$$

In order to estimate the components the first step is to fit a trend curve (linear, polynomial, etc) and remove it from the data by either subtracting (additive model) or dividing (multiplicative). The resulting series approximates $S_t + u_t$, which again is approximated to determine the seasonal component. Finally, after removing both the trend and seasonal component, ideally only the random noise remains. Testing if the remainder is in fact random, is a good indicator, that most of the information of the series is extracted (Hyndman and Athanasopoulos, 2013). Figure 3 shows an example for a multiplicative decomposition on the airline passenger dataset from Box et al. (1970).

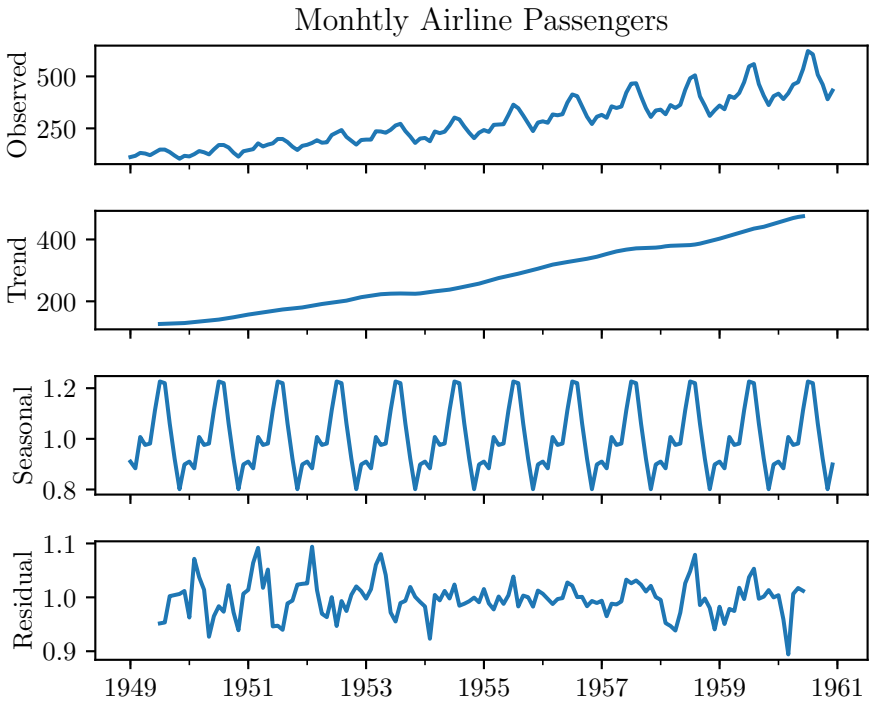

Figure 3: Time series decomposition with *statsmodels*

2.1.2 Exponential Smoothing

Later in the 1950s and 60s, Brown (Brown, 1959), Holt (Holt, 1957) and Winters (Winters, 1960) developed the method of exponential smoothing for time series forecasting. Predictions based on these methods are weighted averages of previous observations, where the corresponding weight decreases exponentially with the observations age.

$$\hat{y}_{T+1} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \dots \tag{2.4}$$

The simplest form of exponential smoothing, simple exponential smoothing (SES), is suited for time series without trend or seasonality. In *component form* it can be written as

$$\begin{aligned} &\text{Forecast Equation} && \hat{y}_{t+1} = \ell_t \\ &\text{Smoothing Equation} && \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1} \end{aligned} \tag{2.5}$$

(Hyndman and Athanasopoulos, 2013). To make predictions, the parameters for smoothing factor (α) and initial value (ℓ_0) need to be chosen, usually by minimizing the squared error using an optimiziation algorithm, since it is a non-linear minimization problem.

Subsequently, exponential smoothing was expanded by Holt (1957) and Winters (1960) to what is now known as the "Holt-Winters seasonal method". The method consists of one forecasting equation and three smoothing equations, one each for the level ℓ_0, the trend b_t, and the seasonal component s_t, with their respective smoothing factors α, β and γ. Here, the letter m is used to denote the frequency of the seasonality, e.g. 12 for a yearly seasonality in a monthly time series. In the additive method, which is used when the seasonal variation is constant over time, the seasonal component is expressed in absolute terms and subtracted in the leveling equation in order to seasonally adjust the data.

$$\begin{aligned} &\text{Forecast Equation} && \hat{y}_{t+1} = \ell_t + b_t + s_{t+1-m} \\ &\text{Level Equation} && \ell_t = \alpha\left(y_t - s_{t-m}\right) + (1-\alpha)\left(\ell_{t-1} + b_{t-1}\right) \\ &\text{Trend Equation} && b_t = \beta^*\left(\ell_t - \ell_{t-1}\right) + \left(1-\beta^*\right) b_{t-1} \\ &\text{Seasonal Equation} && s_t = \gamma\left(y_t - \ell_{t-1} - b_{t-1}\right) + (1-\gamma)s_{t-m} \end{aligned} \tag{2.6}$$

2.1.3 Autoregressive Integrated Moving Average Models

Autoregressive integrated moving average models (ARIMA) aim to describe autocorrelations of the data, rather than trend and seasonality (Hyndman and Athanasopoulos, 2013). Popularized by Box et al. (1970), ARIMA models have since been extended and adapted considerably in many areas of science (De Gooijer and Hyndman, 2006). Generating an ARIMA-model usually consists of three steps:

1. **Obtaining a stationary series (differencing)**: Since an ARIMA models describe a random process, the time series needs to be stationary, which it is not if it has either a trend or seasonality. The trend is usually achieved by repeatedly *differencing* the series with a lag of 1: $\Delta y_t = y_t - y_{t-1}$. The number of repetitions is denoted by the hyperparameter d. Seasonality is removed by differencing not with the neighboring value, but instead the value one season apart, e.g. $\Delta y_t = y_t - y_{t-7}$ for daily data with weekly seasonality. When using the model for predictions, the process has to reversed by *integrating* accordingly.

2. **Autoregressive component**: The autoregressive component makes predictions based on the previous values. The result of this step is an equation for the weighted sum of previous values:

$$y_t := c + \alpha_1 y_{t-1} + \ldots + \alpha_p y_{t-p} \tag{2.7}$$

   where p is the number of previous values used and a hyperparameter. The parameters α_i are approximated using linear regression. Increasing the number of lags used (p) might lead to a better fit on the learning data, but also increases the likelihood of overfitting (Hyndman and Athanasopoulos, 2013). While the choice of optimal hyperparameters used to be a judgment call of the data scientist, there are now plenty of methods for estimating optimal hyperparameters (De Gooijer and Hyndman, 2006).

3. **Moving average – Prediction using previous errors**: A moving average model of order q describes a random process, that only depends on white noise and the weighted average of the q previous values of itself. Assuming the autoregressive component is able to approximate the series, the remaining

error should look like white noise, and the series can therefore be approximated by:

$$y_t := \text{AR-Component} + b_{t-1} \cdot \text{Error}\,(y_{t-1}) + \ldots + b_{t-q} \cdot \text{Error}\,(y_{t-q}) \tag{2.8}$$

The result of these three steps is an **ARIMA**(p, q, d) **model**. In order to describe seasonal data more accurately, there have been extensions to ARIMA, that not only consider the previous observations, but also observations at previous seasons (Williams and Hoel, 2003). Such a seasonal ARIMA model is additionally defined by the length of a season S, as well es the number of seasonal orders (P, Q, D): SARIMA$(p, q, d)(P, Q, D)_S$). However, underlying assumptions and the overall approach remains unchanged.

Lastly, ARIMA models that also consider exogenous variables are called ARIMAX models (Autoregressive Integrated Moving Average with Explanatory Variable). The explanatory term is similar to the other two, in that it is also a weighted sum of the previous explanatory variables x_k:

$$y_t = c + \sum_{i=1}^{p} a_i y_{t-i} + \sum_{j=0}^{q} b_j u_{t-j} + \sum_{k=0}^{n} d_k x_{t-k} \tag{2.9}$$

2.1.4 Forecasting using Neural Networks

Artificial neural networks are an approach for supervised machine learning tasks loosely inspired by the mechanisms of the brain that can be applied to both regression and classification tasks. It can be understood as a "network of 'neurons' which are organized in layers" (Hyndman and Athanasopoulos, 2013). For this work the focus will be on the multilayer perceptron (MLP). A MLP consists of an input layer, an output layer, and at least one hidden layer. The input layer simply passes the input vector on to every perceptron in the first hidden layer. The perceptron is the smallest unit of the network and inspired by synapses of the brain. Mathematically, it will compute the weighted sum of the inputs given ($S = b + \sum_{i=1}^{n} X_i w_i$), including a bias b, to which it then applies a function F called the activation function ($Y = F(S)$), which will produce the final signal, that is sent to the next layer. The values of the weights and the biases are adjusted during learning. The next layer can either be another hidden layer or the final output layer, which is again made

of perceptrons. In case of a classification task there usually is one perceptron per class, whereas a MLP for regression tasks ends in a single perceptron.

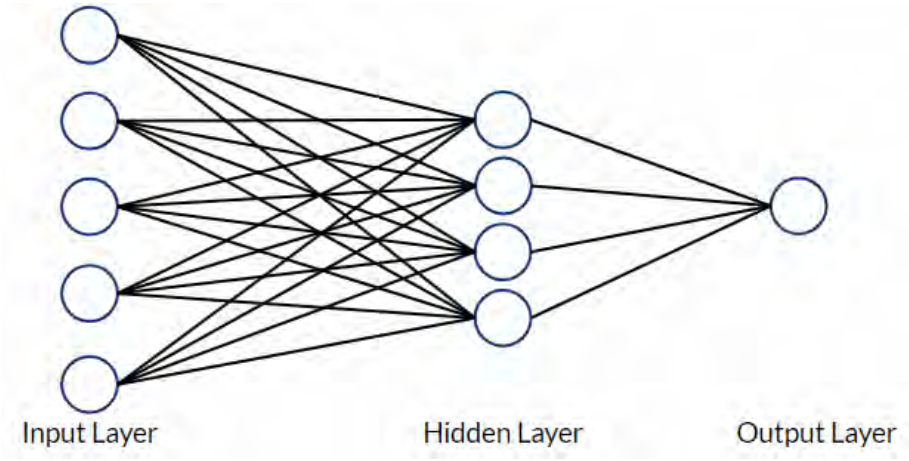

Figure 4: A neural network with five input neurons and one hidden layer with four neurons

If all neurons use a linear activation function, the whole network will act as a linear transformation, which is usually not desired. Usually, a sigmoid function is used, such as the hyperbolic tangent $y(v_i) = \tanh(v_i)$ with a return value ranging from -1 to 1 or the logistic function $y(v_i) = (1 + e^{-v_i})^{-1}$, which is similar in shape but with a range of 0 to 1.

For a MLP with a single hidden layer and a sigmoid activation function, Lewicki and Marino (1989) proved that it can approximate any continuous function with a finite number or neurons (the so-called "universal approximation theorem"). The mathematical assumptions under which this theorem holds are still an active area of research, especially when it comes to deeper networks with more hidden layers and computationally simpler activation function (Lu et al., 2017). One simpler activation function is the rectifier $f(x) = x^+ = \max(0, x)$. Its design is again inspired by neurons (which cannot fire a negative electric impulse), but primarily motivated by the computational complexity of aforementioned sigmoid functions. Not only is the value easier to compute, it also allows for cf, leading to fewer computations overall.

Most methods used for training the model are based on the error back propagation algorithm – iteratively adjusting the weights in order to decrease the error on the training set. The algorithm can be described by the following three steps:

1. **Initialization**: A network architecture (number of layers and neurons) and

the activation functions are chosen, after which the weights need to be initialized, which are usually drawn from a uniform distribution with zero mean and low values.

2. **Forward Computation**: All input vectors are passed through the network and each neuron applies its weighted sum and activation function. At the end an error signal is computed. In case of a regression task this could be the squared distance of the single output signal to the real signal.

3. **Backward computation**: In order to reach a (local) minimum of the error produced by the network weights are adjusted based on the error signal. The amount of adjustment can usually be adjusted by hyperparameters such as the learning rate or the momentum, which need to be adjusted to achieve a balance between learning speed and stability.

When predict values of a time series, the problem needs to be reframed as a regression problem. One such approach is a neural network autoregression model or NNAR model (Hyndman and Athanasopoulos, 2013). It uses the lagged values of the time series as input and the next observation as the output, the relation between which has to be learned. When applied to seasonal data, the NNAR$(p, P, k)_m$ model is described by four parameters:

- p: The number of lagged (previous) values that are part of the input
- P: The amount of previous seasons to be included
- k: The number of hidden neurons in the single layer
- m: The length of one season, e.g. 12 for monthly data

For example a $NNAR(3, 2, 4, 12)$ has a total of four inputs (the three previous observations and two observation from the last two season), four hidden neurons and a seasonal length of 12 (monthly data). Compared to a SAMIRA model, which would "see" the same values, NNAR has two main advantages. Firstly, the series does not need to be stationary. Secondly, the NNAR model is non-linear and might therefore uncover non-obvious relationships.

The NNAR model can also be expanded to include any arbitrary additional information (e.g. day of the week, temperature) simply by expanding the input vector,

and maybe changing the structure of the hidden layer(s) to balance learning performance and accuracy. Often these other values are on a different scale as the time series, which can make learning more difficult. "One of the most common forms of preprocessing consists of a simple linear rescaling of the input variables." (Bishop, 1995, p. 198), usually between 0 and 1 (scaling) or standardized with zero mean and a standard deviation of 1 (standard scaling).

2.2 Concept Drift

2.2.1 Definition

Nowadays, learning algorithms are often deployed online and required to make continuous predictions or classifications (Chen et al., 2012). One example is a credit card company monitoring all transactions in order to classify them either as "fraudulent" or "honest". The main assumption is that both past and future transactions are generated randomly from the same probability distribution. The algorithm is therefore able to differentiate between the two classes based on previous examples and the patterns it derived from them (Gama et al., 2004a). However, spending habits of a person might change over time because of new interests. Or, the location of transactions might change due to a relocation. At the same time, it is safe to assume that fraudsters will always try to adopt their behaviours to remain undetected by the algorithm (Malekian and Hashemi, 2013). As a result the relationship between the observations and the labels (*fraudulent* and *honest*) may change compared to the training set.

In literature, concept drift is often fundamentally distinguished in two types: *real* and *virtual* drift (Widmer and Kubat, 1996). *Real* concept drift describes a change that happens in the real world (like a behaviour change), which changes the relationship between the input and the output. Whereas *virtual* concept drift occurs, when the distribution of the incoming data changes, without an impact on the intrinsic relationship between input and output (Tsymbal, 2004). So far, most studies have looked at concept drift in classification settings, compared to regression (Baier et al., 2019). Therefore the following section will follow the notation of classification problems.

Formally, in Bayesian decision theory, an instance is classified based on a maximal a posteriori probability. Instances are described by a p-dimensional feature vector X and y is the corresponding label. Together they form a labeled instance (X, y). For a given label y the a posteriori probability is given as

$$p(y|X) = \frac{P(y)p(X|y)}{p(X)}. \tag{2.10}$$

Any of these distributions can be a source for concept drift (Žliobaitė, 2010), and be of different concern in any given task. A change in $p(y|X)$, usually as a result of a change in the class conditional probabilities $p(X|y)$, is referred to as *real concept drift* and requires adaptation in the learning algorithm. Class prior probabilities $P(y)$ may change without effecting the decision boundary. But in cost-sensitive scenarios, where different kinds of errors (false-positives, false-negatives) have different associated costs, this might still lead to different decisions. Similarly, a change in the distribution of incoming data $p(X)$ can occur without affecting $p(y|X)$, which is called *virtual concept drift*.

More succinctly, as defined by Gama et al. (2014), concept drift occurs between two time points $t_0, t_1 \in \mathbb{N}$, if there exists a X such that

$$p_{t_0}(X, y) \neq p_{t_1}(X, y), \tag{2.11}$$

where p_{t_i} describes the joint distribution at time t_i of the input and output variables for $i = 0, 1$. Since the joint distribution is comprised of the aforementioned distributions, these two definitions are functionally identical.

Besides the classification of concept drift in *virtual* or *real*, there are other commonly used ways of describing and differentiating different occurrences of concept drift.

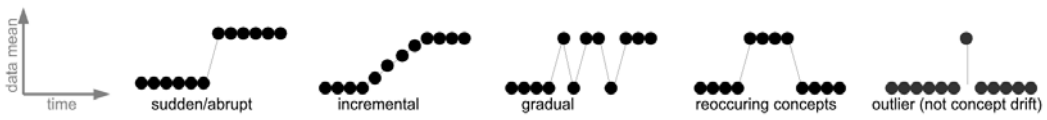

Figure 5: Types of concept drift (Gama et al., 2014)

One such categorization is how the drift develops over time. Gama et al. (2014) distinguishes between four categories, illustrated in Figure 5. A *sudden* concept

drift occurs instantly from one observation to the next (e.g. using a new sensor in a predictive maintenance scenario), whereas *incremental* and *gradual* changes happen over a longer period of time. An *incremental* concept drift is a continuous shift towards a new concept with many steps in between (e.g. sensor wear and tear leading to less accuracy). During a *gradual* change observations from two concepts can be made at the same time, while the probability of finding an observation from the old concept eventually reaches zero.

Concept drifts can also be *reoccurring*, where a concept might reappears at a later point in time (e.g. fashion trends, that seem to reappear arbitrarily). According to Žliobaitė (2010) this differs from the common notion of seasonality, since "it is not certainly periodic". Finally, it is important to not label an outlier, or plain random noise, as a concept drift, since they do not require an adaption.

2.2.2 Handling Concept Drift

Table 1 gives an overview on strategies which can be applied for detecting and handling concept drift. The first dimension refers to the application of an explicit drift detection algorithm. The second dimension describes the adaptations of the underlying ML model. The simplest option is the development of a robust, static ML model which is trained once and then deployed for an ongoing prediction (Guajardo et al., 2010), the upper left case in Table 1. Other approaches continuously adapt the prediction model, e.g. with a sliding window where new data instances are continuously used to adapt the prediction model (Kuncheva and Žliobait\.e, 2009). Such approaches rely on an ongoing adaptation of the prediction model. The frequency of the adaptation can be chosen arbitrarily, or be based on expert knowledge, like known seasonal patterns (Guajardo et al., 2010). Depending on the complexity of the model, this requires a lot of computational power. Furthermore, time constraints might also not allow the retraining of the entire model before the next prediction is required, especially in environments with limited resources, e.g. on mobile devices (Oneto et al., 2015). The lower part of the table depicts approaches which rely on a dedicated drift detection. Drift detection can be handled by an algorithm which detects drifts in the incoming data or the distribution of the prediction error. Based on detected drifts, the model can either be retrained (Pechenizkiy et al., 2010) or

another model can be applied. Approaches with and without drift detection are also referred to as *implicit* and *explicit* detection models (Cavalcante et al., 2016).

		Model Adaption	
		No	Yes
Drift Detection	No	Static model	Window-based approach
	Yes	Drift detection with model change	Drift detection with model adaptation

Table 1: Model adaptation and drift detection options

Explicit drift detection methods can either monitor incoming data streams or the error-rate of base learners (Cavalcante et al., 2016). These methods detect concept drifts using statistical tests. Their main advantage over implicit methods lies in their relative resource efficiency and their white-box approach, where the user is informed about the presence of a concept drift. Looking back at the definition for a posteriori probability (see equation 2.10), error-based approaches only consider changes in a posteriori distribution $p(y|X)$ through the changing classification error. On the other hand, detection-based methods observing incoming data streams look only at $p(X)$, which in turn can mean that a virtual concept drift is detected, which does not actually have an impact on the classification task.

A well-known drift detector is the **drift detection method (DDM)** (Gama et al., 2004b). As an error-based approach, it tracks the classification error of a base learner and assumes that a change in concept leads to more frequent miss-classifications, based on the PAC learning model (Probably Approximately Correct). As new classifications are made, the detector computes the rate of miss-classification so far p_i and the standard deviation ($s_i = \sqrt{p_i(1-p_i)/i}$), storing the values as p_{min} and s_{min}, when $p_i + s_i$ reaches its minimum. The error-rate p_i is a random variable from Bernoulli trials and can be described by a Binomial distribution, which for a sufficient number of samples ($n > 30$) is approximated by a Normal distribution. A drift warning level is reached, when $p_i + s_i \geq p_{\min} + \alpha \cdot s_{\min}$. The authors suggest a warning level of $\alpha = 2.0$, which corresponds to a confidence of 95%, that a change in distribution has occurred. Once the warning level is reached the detector stores the incoming instances in anticipation of a possible concept drift. If

$p_i + s_i \geq p_{\min} + \beta \cdot s_{\min}$, the drift level is reached. The suggested threshold of $\beta = 3.0$ indicates 99% confidence, that a drift has taken place. Consequently, the detector and base learner are reset, and a new base learner is trained using the instances stored since the warning level. The factors for the warning and drift level can be changed to adopt the drift detector to the given context.

Expanding on this, Baena-García et al. (2006) developed the **early drift detection Method (EDDM)**, which is more geared towards detecting slow gradual changes by using the distance-error-rate p_i of the base learner, which describes the distance between two classification errors. Instead of a decreasing error-rate, the EDDM expects the distance-error to increase over time, storing p_{max} and s_{max} when $p_i + 2 \cdot s_i$ reaches its maximum value. Again, there are two thresholds, determined by the parameters α and β. A warning level is reached, when $(p_i + 2 \cdot s_i) \,/\, (p_{\max} + 2 \cdot s_{\max}) < \alpha$, and a drift level when $(p_i + 2 \cdot s_i) \,/\, (p_{\max} + 2 \cdot s_{\max}) < \beta$. In the original paper α and β are set to 0.95 and 0.90 respectively.

The **ADaptive sliding WINdow (ADWIN)** detector (Bifet and Gavaldà, 2007), works without a base learner and instead evaluates the averages over dynamically sized windows. When the mean remains relatively constant, the window size increases up until a maximum size W. However, it uses a variant of the exponential histogram to store the data in only $O(\log W)$ memory. Within this adaptive window, the detector tries to find two *large enough* sub-windows that are *distinct enough*, and if it does, the instances of the sub-windows are drawn from different distributions, and a concept drift is detected. The detector can be tuned with the hyperparameter δ, called the confidence parameter, which quantifies the threshold for "distinct enough". While ADWIN does not assume any particular data distribution, its main disadvantage is that it does not detect gradual concept drift as good as other approaches (Gonçalves et al., 2014).

A detector that can both be used to monitor the incoming data stream directly or through the accuracy of a base learner, is the **Page-Hinkley (PH) Test** (Page, 1954). This sequential analysis technique tracks the average accuracy up to the current point in time $\bar{x}_T$ and the cumulative difference between the average and the actual values $m_T = \sum_{t=1}^{T} (x_t - \overline{x}_T - \delta)$, where δ specifies the tolerable magnitude of changes. PH tests the difference between the current value of m_T to its minimum

observed value and flags a change, when the difference is larger than the threshold λ (user defined).

2.2.3 Concept Drift in Regression Tasks

As motivated in Section 1), this work will focus on a real world dataset describing taxi demand in New York City, with the goal of predicting the future demand. As such, this task is a regression problem, instead of a classification problem which has been the main focus of this chapter so far. Unfortunately, most of the research examined by Baier et al. (2019) focuses on classification tasks (29/34). However, many ML challenges need to be modelled as regression tasks and approaches for classification cannot be applied, e.g. 20 out of 89 studies applying ML and being published in ECIS and ICIS between 2010-2018 are regression problems. But, there is a key difference between classification and regression tasks, when it comes to concept drift, which so far has not been studied extensively: "Every change in $p(X)$ affects $p(y|X)$, since $p(X)$ and $p(y)$ are drawn from the same distribution." (Cavalcante et al., 2016). Therefore the notion of concept drift goes beyond simply determining the trend, as a change in concept can change the overall shape of a time series, without affecting its mean.

Detectors such as DDM or eDDM are designed for classification problems, meaning they assume the classification error to be the result of a Bernoulli process and therefore follow a binomial distribution. The error of a regression problem however, is continuous instead of discrete and follows a nominal distribution. Still, as long as the Central Limit Theorem holds, a Binomial distribution is closely approximated by a Normal distribution for a a sufficient large number of examples, which is also exploited by DDM Gama et al. (2004b). Cavalcante and Oliveira (2015) have adapted DDM for a regression task, monitoring the mean prediction error instead of the classification error-rate. On the other hand, eDDM evaluates the distance between classification errors, which has no direct analogy in a regression setting. For this type of detector it is necessary to transform the prediction error into a classification error, e.g. by labelling predictions that fall within a certain percentage of the real value as correct (Xiao et al., 2019). On the other hand, drift detectors that monitor the data stream directly, such as PH and ADWIN do not need to be changed in a

regression setting.

2.2.4 Commonly used Datasets

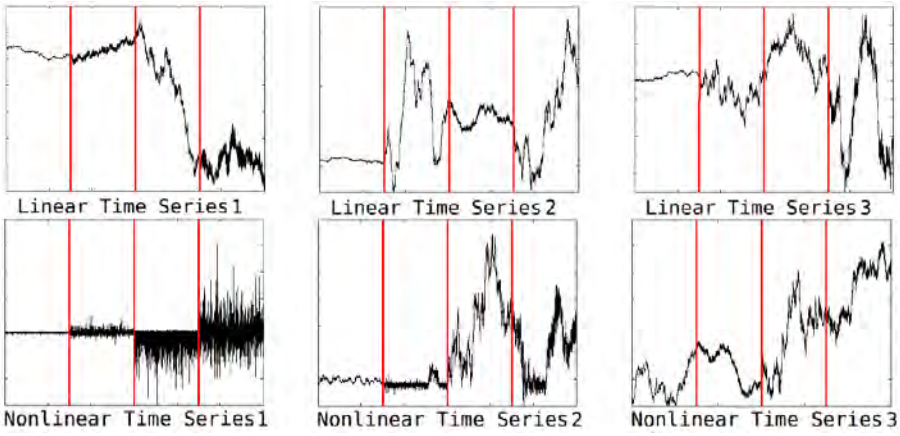

Figure 6: Example of artificial time series used by Cavalcante et al. (2016)

In order to evaluate and compare these different approaches most papers use a combination of artificial and real-world datasets. Using artificial or generated datasets has multiple advantages: Anyone can recreate the dataset, there is no ambiguity regarding preprocessing and it is known when and how strongly concept drift occurs (Gama et al., 2014). All of the artificial datasets presented by Gama et al. (2014) generate labeled data for a classification task. For regression tasks, one approach used by Cavalcante et al. (2016) is to generate time series (e.g. based on an autoregressive process, see Section 2.1), and "stitch" them together, creating a new time series with known concept drifts. A visualization is shown in Figure 6. When evaluating a drift detector in isolation, common evaluation metrics are accuracy, evaluation time, as well as false alarm and miss detection rates (Gonçalves et al., 2014).

While evaluations on artificial datasets have a high degree of internal validity, external validity can only be assumed (Gama et al., 2014). Therefore using real-world datasets is also valuable in determining the effectiveness of concept drift adaptations. However, the main challenge is that most of the real-world datasets used have "little concept drift in them, or concept drift is introduced artificially" (Tsymbal et al., 2008). Among the most commonly used datasets are the Airlines and

Electricity datasets provided by the Massive Online Analysis framework.[1]. In the Airlines dataset each sample represents a single flight and the task is to predict if the flight arrived on time, ranging from 1988 to 2008. The Electricity dataset describes the prices, which are affected by demand and supply. The label corresponds to the price being above or below the moving average of the last 24 hours. It contains 45.312 instances over 2 years.

The aim of this work is to analyze a dataset which shows the following characteristics:

- It is openly available, making it easy to reproduce the results
- Novelty in the context of concept drift research, validating approaches on another real-world dataset
- It describes a regression task, to expand the discussion of concept drift into this area
- It spans a relatively large time span, making it more likely that concept drift occurs

The NYC taxi trip dataset (TLC, 2019) fulfills these criteria. Therefore, it will be subject of the further analysis of this thesis. A detailed description of the dataset can be found in Section 3.1.

2.3 Data Mining Process under Concept Drift

Evolving streams of data challenge many assumptions of standard data mining and machine learning settings (Žliobaitė et al., 2016). In a system that does not consider the changing nature, models may become less accurate over time or "opportunities to improve the accuracy might be missed" (Žliobaitė et al., 2016). Therefore the challenges of concept drift have to be considered at every point in the data mining or machine learning process.

CRISP-DM (cross industry standard process for data mining) is a standardized process for data mining projects. The methodology is described on four *levels* of abstraction: phases, generic tasks, specialized tasks, and process instances (Wirth, 1995). However, for this work, the focus will be only on the six phases:

[1]Theses datasets can be accessed at `https://moa.cms.waikato.ac.nz/datasets/`

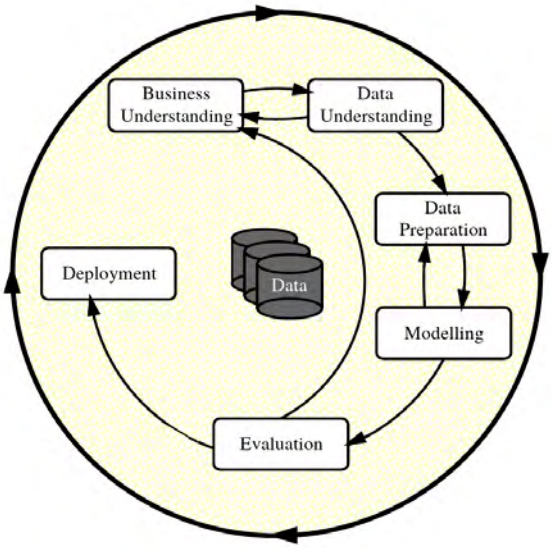

Figure 7: Phases of the CRIPS-DM Process Model for Data Mining (Wirth, 1995)

1. *Business Understanding* aims at determining the business goals and requirements and translate them into a rough outline.

2. *Data Understanding* is about doing a first analysis of the available data and its quality.

3. *Data Preparation* starts with data selection and is often accompanied by building or maintaining a data warehouse. It also includes any form of preprocessing like handling missing information, removing redundancies, or reducing data.

4. *Modeling* focuses on the selection, initialization and tuning of data mining techniques, usually comparing multiple approaches.

5. *Evaluation* typically considers offline evaluation on historic data to determine the best approach for the given business goal.

6. *Deployment* of the final model concludes the phases, when the model is integrated in the existing service environment.

As shown in Figure 7, the process is not linear and contains certain feedback loops. However, it still assumes that most phases are performed offline, separate from the deployed model. As data streams are expected to change, monitoring and updating models should become a "natural and core part of the data mining process" (Žliobaitė et al., 2016). A modification of CRISP-DM for an adaptive setting is

shown in Figure 8. The main difference is, that data preparation, mining, and evaluation is automated as part of the deployment.

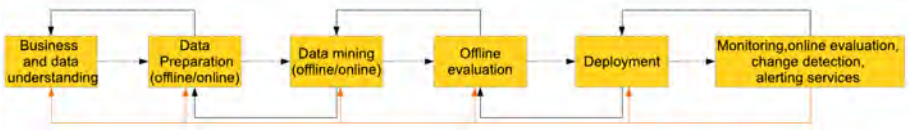

Figure 8: Towards CRISP for adaptive mining (Žliobaitė et al., 2016)

A standardized process like CRISP-DM leaves many questions purposefully open to the practitioner. In their book *Machine Learning Logistics*, Dunning and Friedman (2017) propose an architecture that facilities model management in a streaming context called the "Rendezvous Architecture". An outline of the architecture is presented in Figure 9. It expands the basic request/response paradigm in multiple key areas:

1. *Leveraging streaming-based micro-services*: Having a single application to digest a request, perform preprocessing, do computations and return a value is only feasible in early stages. Separating concerns into encapsulated micro-services allows for more flexibility and traceability of changes. Connecting these services via data streams (such as the open-source library kafka[2]) de-couples the services and allows them to be switched out with minimal impact.

2. *Containerization*: Putting every piece of code inside a container allows it to be executable and behave predictably on both local machines and large server farms. A data scientists does therefore not have to worry about any side-effects and can focus on his work, while orchestrators ensure that the production system is running.

3. *Keeping multiple production models*: There can be plenty of reasons to keep many models online. E.g. when A/B-Testing to evaluate the effects a new model has, or to compare multiple models in a real production environment. Through the de-coupled architecture, this can be done with minimal impact on the service.

4. *Rendezvous Server*: A Rendezvous Server is deployed and operates between the output-stream from the models and the response stream. For every request

[2]https://kafka.apache.org/

it sees, an inbox is opened to keep track of incoming decisions by the models. Here, decisions can be implemented on how to choose which model result to return to a given request.

This decision can be based on A/B-Test requirements, model performance, or be timer-based, returning the best available result after a certain time has passed to fulfill service level agreements.

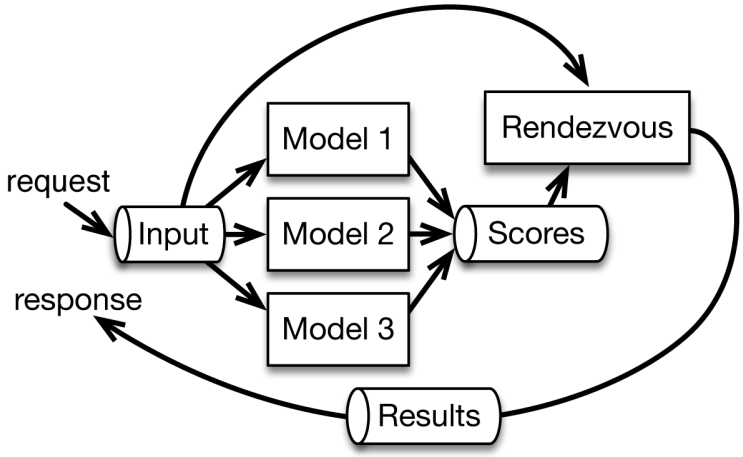

Figure 9: Core rendezvous design (Dunning and Friedman, 2017)

Not all of the models reading from the input stream need to serve the same function. Figure 10 shows how there can be models with a different target than the result stream, where their result would be picked up by the Rendezvous service. A *decoy* model is used solely to archive the data stream. Archiving the data stream here, ensures that data is archived exactly as it was seen by the models. This can help when debugging a problem or doing an analysis later. Additionally, a *canary* model should be deployed. The canary model can be a former production model, which is well-understood, but has since been replaced by a generally improved model. Or it is a model that was specifically designed to be easily understandable, with "good enough" results, like a simple decision tree or a linear regression. There are two main advantages to this approach: First, if the delta between the canary and the real model changes, it can be assumed that the input data has changed. In this way it serves as some kind of drift detector. Second, comparing every new production model to the same canary model helps data scientists build an understanding of its behaviour enabling them to detect anomalies quickly.

In conclusion, it becomes clear that the challenge of concept drift cannot simply be seen a challenge on the algorithmic level. If businesses want to reap the benefits of analyzing changing data streams, the effects of and measures against concept drift need to be considered early on. De-coupling the overall architecture, as done in the Rendezvous Architecture, and thinking of drift detection as one part in a process can help to build a dynamically adaptable and extensible data service.

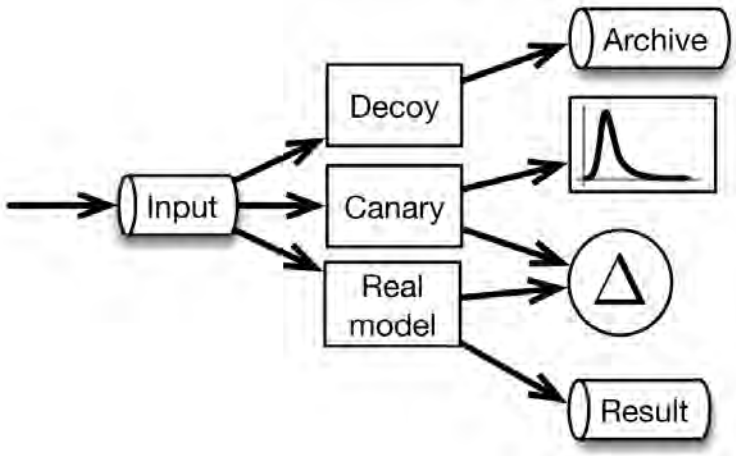

Figure 10: Integration of a canary model into the rendezvous architecture (Dunning and Friedman, 2017)

2.4 Taxi Demand Prediction

Thanks in large part to the momentum of the sharing economy through companies like Uber, Lyft and Lime, predicting traffic patterns and demand in cities has become an increasingly popular area of research (Abduljabbar et al., 2019). These usually app-based businesses are capable of collecting large quantities of precise data. In a fierce market filled with venture capital, they must compute on user experience and comfort, which is often facilitated by machine learning approaches (Liao et al., 2018). This section discusses related work in the research area of short-term taxi demand forecasting, focusing on the New York City taxi dataset. A more detailed look at the dataset, and how it is preprocessed for this work is given in Section 3.1.

One particular area of interest is the question of how to discretize the (almost) continuous feature space into spatial and temporal buckets. Xu et al. (2018) evaluate the data over a 3.5 year period starting in 2013. They divide the location data into a grid with a cell size of about 153mx153m and feed weekly sequences taxi demand

Figure 11: Representation of the Geohash and Voronoi tessellations as graphs, from Davis et al. (2019)

data into a Long Short Term Memory (LSTM) recurrent neural network (RNN). When evaluating the accuracy of their model using different temporal resolutions (from 5 to 60 minutes), they conclude that the "RMSEs are very close under different time-step lengths." (Xu et al., 2018). The question of spatial discretization (also referred to as tessellation) has been analyzed by Davis et al. (2018), comparing Voronoi tessellation with Geohash tessellation. Veronoi tesselation is a grouping based on nearest neighbors, whereas Geohash tessellation is grid based (see Figure 11). The predictions were done using statistics-based models like ARIMA and exponential smoothing. They conclude, that "Geohash tessellation was the winning strategy for [New York City]" (Davis et al., 2018), while noting that the optimal choice of tessellation strategy is heavily dependent on "the demand density in each partition, and the geography of the city". In their follow-up paper, they introduce an ensemble approach, which combines predictions based on each of the tessellations (Davis et al., 2019). Thereby they achieved a prediction accuracy "close to the best model at each time instant".

However, it is not possible to directly compare the error metrics provided in these papers, since the results of deep learning methods are known to be very dependent on the dataset, in this case on the exact preprocessing. Especially the size of each cell, and therefore the amount of demand in each cell has a large impact on the error metrics. This will be discussed in more detail in Section 3.2.4. An apples-to-apples comparison between two deep-learning approaches was made by Liao et al. (2018) on the NYC taxi dataset. The first approach, called ST-ResNet, is built around the convolutional neural network ResNet and was originally applied to ride-hailing data from Beijing (Zhang et al., 2016). The core idea was to model the

problem in terms of inflow and outflow of crowds into each region, where people might want to leave a region again via taxi after a certain amount of time has passed. Three separate networks were trained to capture this dependency at short-, medium-, and long-term temporal distances and the predictions of each network combined (fused) as a weighted sum. An overview of the architecture is shown in Figure 12. The second deep-learning approach evaluated is called FCL-Net. It has a similar overall structure, learning separate models at three different temporal categories, but instead of ResNet, Ke et al. (2017) use a convolutional long short term memory architecture (ConvLSTM). Their work was originally evaluated on ride-hailing data from DiDi Choxing, a Chinese service operating in Hangzhou.

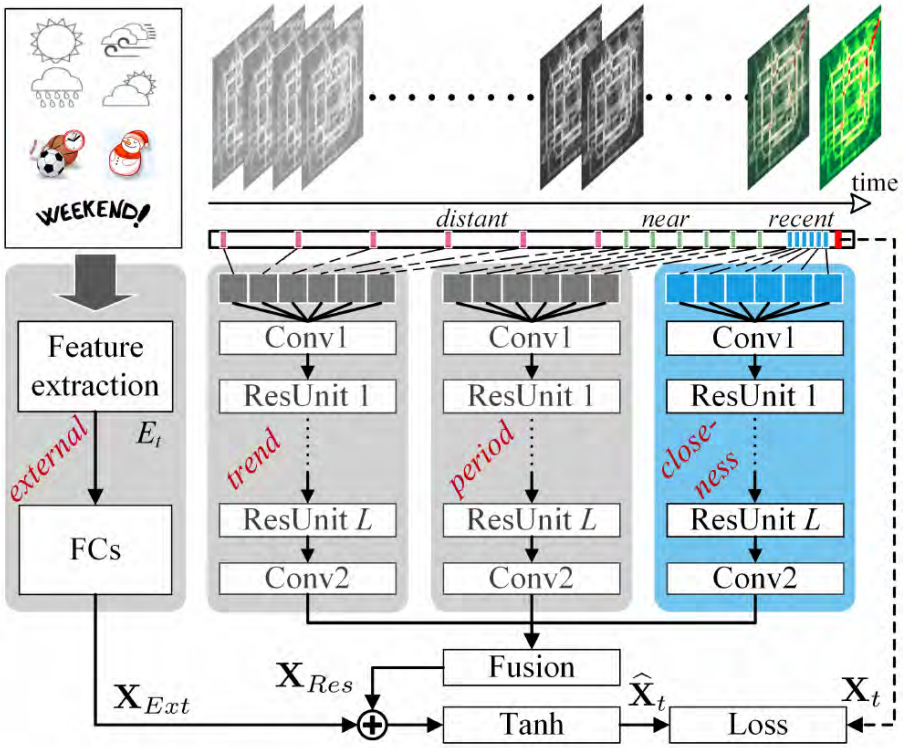

Figure 12: Architecture of ST-ResNet (Zhang et al., 2016)

For the comparison Liao et al. (2018) used two years of data from 2014 to 2016 from Manhattan only, dividing the borough into a 32x32 grid, each cell covers about 200 by 400 meters. Additionally, they annotate the data with information about workdays/weekends, holidays, and 9 weather-related features. Surprisingly to the authors, the supposedly more capable FCL-Net performed worse with an RMSE of 15.11 compared to ST-ResNet's 11.13. According to the authors this could have two possible explanations. Either, the coarse-grained temporal dependencies captured

by the architecture (i.e. using three separate networks) might be more important than the "fine-grained temporal dependencies as captured by LSTM" (Liao et al., 2018). Or the spatial dependency is more important, which is presumably better captured by ResNet.

A drift-aware approach to the taxi demand problem, called BRIGHT, was published by Saadallah et al. (2018). BRIGHT is an ensemble-based approach, which combines a diverse set of models to handle distinct types of concept drift, that might occur in the traffic demand domain. The ensemble operates in two stages: model selection and stacking through model averaging, as illustrated in Figure 13.

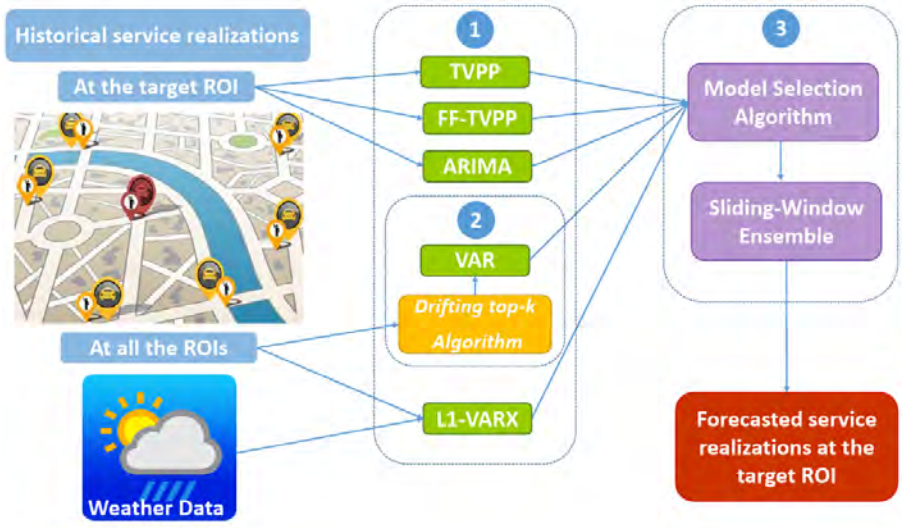

Figure 13: Illustration of BRIGHT framework (Saadallah et al., 2018)

Model selection is done for two *families* of base learners, ensuring a diverse ensemble. A family is a grouping of models that are similar in their approach. The first family consists of univariate models, considering only the values of one time series, i.e. the demand pattern of one region. The univariate family consists of Time-Varying Posson Process (TVPP), Fading-Factor TVPP, and ARIMA. The second family of base models groups multivariate approaches that also consider demand information from neighboring regions. They are an L1-regularized Vector AutoRegressive process with exogenous variables (L1-VARX), and a Drift-Aware VAR process, both of which are novel approaches. For each family a *family winner* is selected based on the a fixed-size sliding window loss. The selection is dynamic and will be re-evaluated, when a Page-Hinkley test triggers a drift alarm.

The second stage, *Stacking through model averaging* combines the two predictions

from the two family winners. Inspired by work from Moreira-Matias et al. (2013b), it uses a fixed-size sliding window to determine the importance of the models in the final decision based on the recent loss. This strategy ensures a "a blind adaptation to drift" (Saadallah et al., 2018) at the second ensemble stage.

The authors validated their approach using datasets from the cities of Porto, Shanghai and Stockholm, as well as synthetic data. They conclude that the approach was in fact capable of adapting to different kinds of concept drift, leveraging the strengths of the individual predictors. However, they note that the simplicity, both in the ensemble approach as well as the individual models, limits the approaches' ability to capture non-linear dependence structures or recall reoccurring concepts.

While there has been a considerable amount of research on this particular dataset, most of it is focused on getting the best predictive performance, and only considers a relatively short period of times. So far, no comprehensive evaluation on concept drift of NYC taxi trips has been published. More recently, some work has been done to consider transportation demand data as an interesting dataset in regards to concept drift.

2.5 Evaluation Metrics

The decision, which measure to use when evaluating a time series prediction, is always a subject for debate (Flores, 1986). The relevant error metric depends heavily on the application, for example when the cost of over-predicting is much larger than the cost of under-predicting. In general, measures for accuracy are divided into relative and absolute measures, the most popular of which are listed in Table 2. In this table the error in period i is defined as the difference between the real value and the forecast, $E_i = X_i - F_i$; the percentage error PE_I for period i is defined as $PE_i = [(X_i - F_i)/X_i] * (100)$.

The meaning of a certain absolute error is uncertain without knowledge of the domain – a deviation of 100 might be acceptable when predicting sales units, but not so when predicting the temperature. Usually, the root mean squared error (RMSE) is used, because it is desirable to weigh outliers more heavily, and the result will have the same unit as the input data (Flores, 1986).

	Name	Equation
Absolute	Mean Absolute Deviation	$\text{MAD} = \sum_{i=1}^{n} \lvert E_i \rvert / n$
	Mean of Squared Errors	$\text{MSE} = \sum_{i=1}^{n} E_i^2 / n$
	Root Mean Squared Error	$\text{RMSE} = \left[\sum_{i=1}^{n} E_i^2 / n\right]^{1/2}$
Relative	Mean Percentage Error	$\text{MPE} = \sum_{i=1}^{n} PE_i / n * 100$
	Mean Absolute Percentage Error	$\text{MAPE} = \sum_{i=1}^{n} \lvert PE_i \rvert / n * 100$
	Symmetric Mean Absolute Percentage Error	$\text{sMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{\lvert F_t - A_t \rvert}{\lvert A_t \rvert + \lvert F_t \rvert}$

Table 2: Commonly used measures for accuracy

Relative errors are easier to interpret, since no prior knowledge of the scale of the input is required. While the mean absolute percentage error is easy to understand and analogous to the RMSE, it has one disadvantage – its value is not bound, and can reach very high values, when the actual value is low. The symmetric mean absolute percentage error (sMAPE) fixes this shortcoming and is always between 0% and 100%. As with any relative measure, problems arise when a series contains zero values, which tends to be the case more frequently when describing demand or count data. If both prediction and actual value are zero, the sMAPE is undefined. Since the prediction was in fact correct, this case needs to be considered and the sMAPE defined to be 0%.

When comparing the accuracy of two models on the same dataset it can be helpful to consider statistical tests to determine, if the difference is significant or simply a result of the specific data values in the sample. In this work the *Diebold-Mariano test* (DM-Test) will be used, as proposed by Diebold and Mariano (1995). It considers the loss-differential d_i of the two forecasts, which is a time series derived from the residuals. Let e_i and r_i be those residuals, and d_i the loss-differential.

$$d_i = e_i^2 - r_i^2 = (y_i - f_i)^2 - (y_i - g_i)^2 \tag{2.12}$$

It is assumed, that the expected loss-differential is equal to zero, e.g. $\mu = E[d_i] = 0$, which forms the null hypothesis. The DM statistic is then defined using the mean differential and the autocorreclation function y_k, which describes the autocovariance at lag k. The number of lags considered, h, can be chosen freely, but it is generally

sufficient to use $h = n^{1/3} + 1$

$$\gamma_k = \frac{1}{n} \sum_{i=k+1}^{n} \left(d_i - \overline{d}\right) \left(d_{i-k} - \overline{d}\right) \tag{2.13}$$

$$DM = \frac{\frac{1}{n} \sum_{i=1}^{n} d_i}{\sqrt{\left[\gamma_0 + 2 \sum_{k=1}^{h-1} \gamma_k\right] / n}} \tag{2.14}$$

If the null hypothesis holds true, the DM follows a standard normal distribution under the assumption that the loss differential time series d_i is stationary. Thus, the two-tailed critical value for the standard normal distribution can be used at the desired significance value α. For $\alpha = 5\%$, the critical z-values are 1.96 and -1.96.

3 Design and Implementation

In this chapter all the moving parts required to evaluate and possibly answer the two research questions are laid out. Section 3.1 builds up an understanding of the taxi demand use-case and the New York City dataset, which will be used throughout the work. Next, the models used for prediction are introduced in increasing complexity and evaluated over a one-year period in 2012. Finally, possible strategies for mitigating the effects of concept drift are presented, including the novel error intersection approach.

3.1 New York City Taxi Demand

The NYC taxi trip dataset (TLC, 2019) is a public dataset provided by the New York Taxi and Limousine Commission (TLC), which is the agency responsible for regulating taxis and for-hire vehicles in the city of New York. It contains information about all individual taxi trips that take place in the city, published separately for yellow taxis, green taxis and for-hire vehicles. Yellow taxis (officially called medallion taxis), are limited by the TLC through the number of licenses made available and are the only type of taxi, that is allowed to be hailed in Midtown Manhattan, Downtown Manhattan and the two airports LaGuardia and JFK. On the other hand, green taxis were introduced in 2011 to better serve the other boroughs of New York. Even if they cannot pick up passengers in the aforementioned regions, they are allowed to drop off passengers there. Finally, for-hire vehicles perform pre-arranged trips and include for example black cars, luxury limousines and app-based services, such as Uber and Lyft.

One goal of this thesis is to explore concept drift on a real world dataset. In comparison to the datasets introduced in 3.1, the NYC taxi trip dataset stands out. It spans a comparatively large time span (over ten years), starting in January 2009, and still is updated regularly. Over time, the dataset has grown to include over 1.4 billion rides by June 2018. Given that taxi trips are closely tied to people's habits it can be assumed, that there is a lot of potential for change over time. At the same time, it seems impossible to also monitor all possible factors, which can impact taxi

ridership. Some, like current weather and public holidays, are relatively easy to include, while popular night clubs, or local subway interruptions are not.

Over time, the data provided by the TLC has changed considerably. Every trip until June 2016 is recorded with precise coordinates for both the drop-off and pick-up locations. This has been widely criticized by data privacy advocates. For once, precise coordinates (reported with 13 significant digits), can directly point to a single household in sparse communities. Additionally, Tockar (2014) showed, how they could link public sightings of celebrities entering cabs with a single data point, where they could look up the tip amount. Since June 2016, the data refers only to the "Taxi Zone" of the pick-up and drop-off, of which there are 263. These zones are not chosen arbitrarily, but instead are roughly based on NYC Department of City Plannings Neighborhood Tabulation Areas, which are meant to approximate existing communities and neighborhoods. Additionally, taxi zone 264 represents coordinates that were not available ("NA") and taxi zone 265 represents rides where the recorded coordinates were invalid ("NV").

Figure 14 shows how the starting points for yellow taxis are distributed over the regions. Because of their special legal status, it makes sense, that the majority of yellow cab pickups take place in Manhattan and at JFK and LaGuardia airport. In fact, of the fifty busiest regions forty-eight are in Manhattan, the other two being the airports. Therefore, this work will primarily focus on the twenty taxi zones with the highest overall demand (which already account for almost 60% of taxi demands), in order to reduce the amount of computational complexity. A list of these regions can be found in Appendix A.1.

In order to use the entire temporal range of data from the dataset, this work will focus on the yellow taxi data, matching the rides before mid-2016 to their corresponding taxi zones. While limiting the scope means that the data does not properly reflect overarching transportation trends of New York City, it more closely resembles a use case, where data on the activity of a competitor is not available. The yellow taxi trip data consists of monthly CSV files, where each line represents a single taxi trip. It includes seventeen attributes per trip, describing the trip itself (pick-up, drop-off, trip distance, passenger count) as well as the fare (amount, payment type, tip/tolls amount). However, since the task at hand is about planning and dispatch-

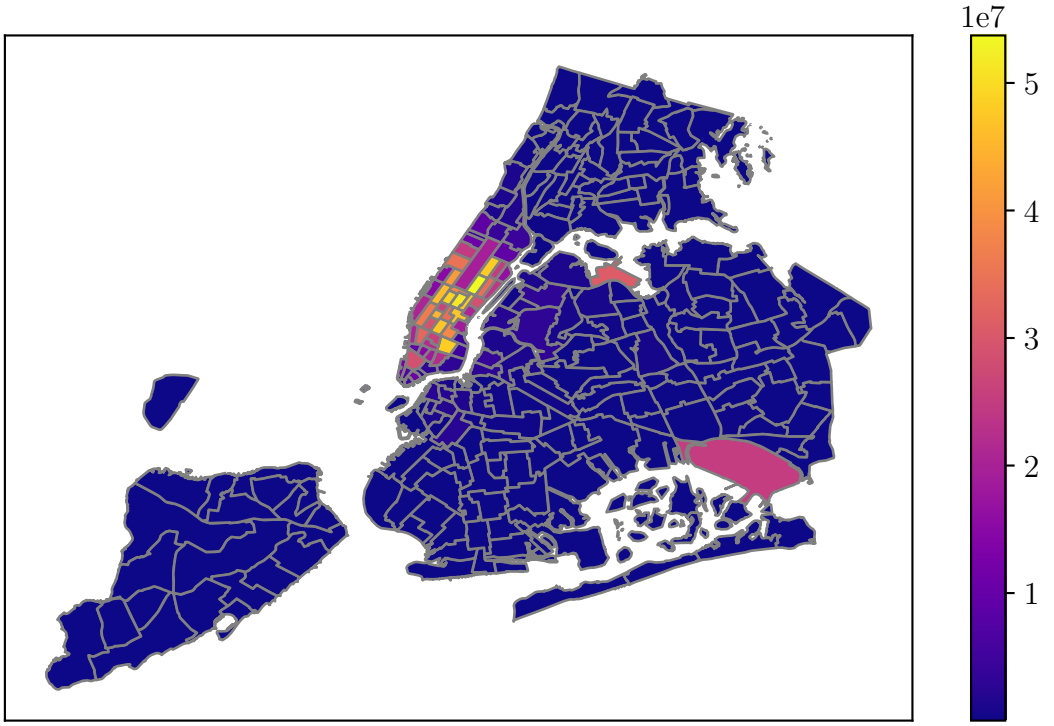

Figure 14: Distribution of rides over regions

ing the taxi fleet, most of these features are not needed. Taxi demand, for this work, will be assumed to be approximated by pick-ups, which is the best available predictor in this case (Liao et al., 2018). Consequently, each trip is characterized only by its pick-up location and timestamp.

As stated by the TLC, the dataset is not cleaned or checked before being published. Therefore, it is necessary to check the data for invalid or implausible records. The following criteria have been used to label a record as invalid and exclude it from further analysis:

1. Trips with exact coordinates that start outside any of the 263 taxi zones
2. Trips that are assigned taxi zones 264 (“NA“) or 265 (“NV“)
3. Trips with a metered distance of zero or lower
4. Trips where the drop-off time is not after the pick-up time
5. Trips with a negative total fare
6. Trips that take place in a different month than the CSV file name indicates

Additionally, the raw data is not aware of daylight saving time, which begins on the

second Sunday of March and ends on the first Sunday of November. This leads to a gap each march when the clocks are moved forward from 2:00am to 3:00am, when working with timezone naïve timestamps, which would violate the equi-distance assumption outlined in Section 2.1. Using a timezone-aware timestamp in *pandas*, this pitfall can be mitigated. The shift back to normal time in November, however, is more challenging. Since the raw data is reported without timezone information the time span between 1:00am and 2:00am contains unusually many rides. For this work the assumption is made that demand is split evenly between "1:00am daylight saving time" and "1:00am normal time".

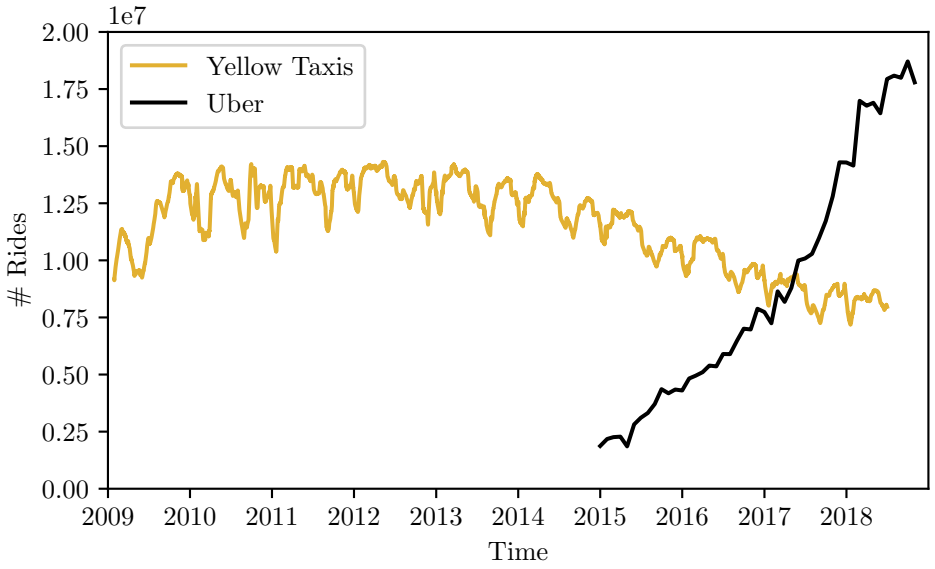

Figure 15: Overall distribution of rides over time, sum of past 28 days

In the field of spatio-temporal forecasting, it is common, and often times computationally necessary, to discretize the data (Davis et al., 2019). Spatially, the data is segmented into the 263 taxi zones for all rides after June 2016. For all rides that took place before June 2016 the coordinates were matched to their respective region. Temporally, the data provides timestamps with an accuracy of one second. Related work using the same dataset has most commonly used an hourly resolution (Liao et al. (2018), Davis et al. (2018), Zhao et al. (2016)), with some also exploring shorter intervals (Xu et al. (2018) or Moreira-Matias et al. (2013a)). Here, the trips will also be binned into hourly intervals.

Figure 15 shows the overall trend of all yellow taxi rides from 2009 until 2018. On this scale, the data seems to follow a yearly seasonality, with local minima around

January and August. After 2012 the overall trend clearly indicates a decrease in demand, presumably due to the rise of competitors. The figure also shows the reported number of rides by Uber (reported as part of the FHV data) increasing rapidly. While Uber has been operating in New York since 2011 (Uber, 2011), the TLC only started providing the data in 2014. In terms of the types of concept drift introduced in Chapter 2.2.1, this can be described as an incremental concept drift where data patterns evolve over time.

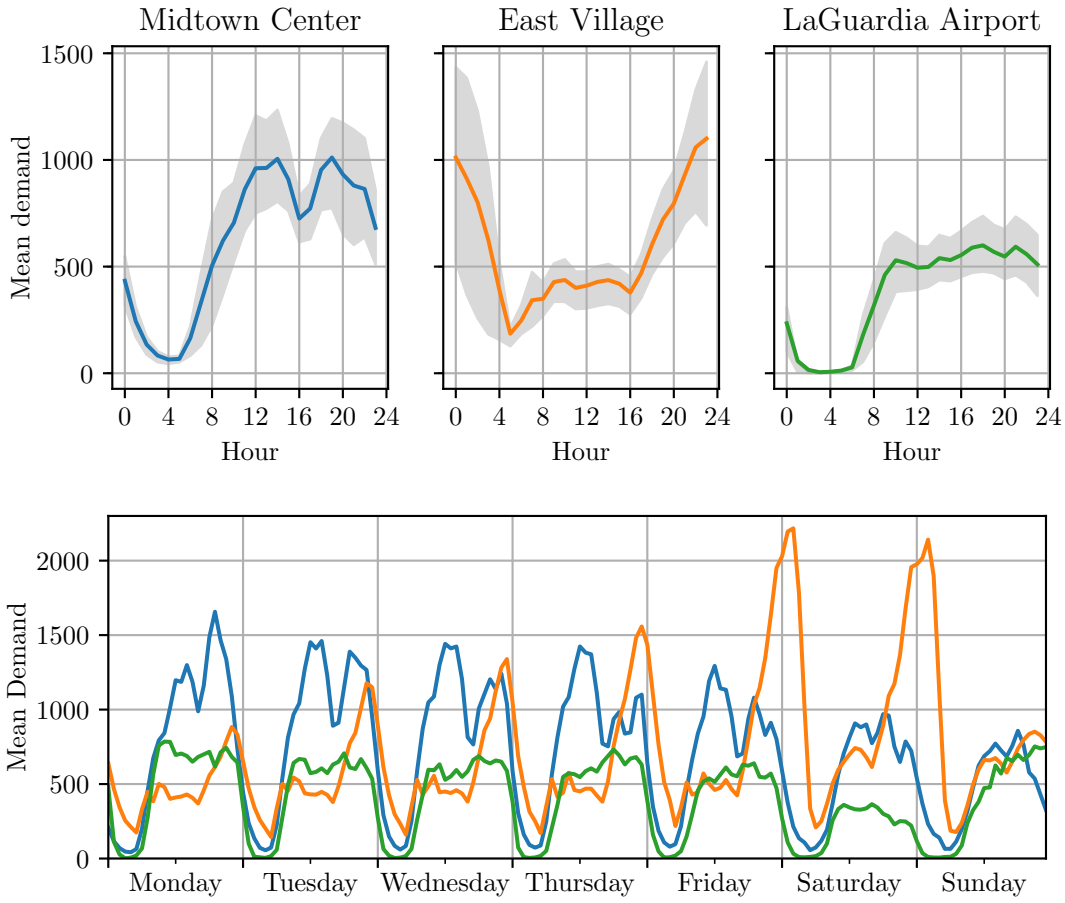

Figure 16: Mean weekly progression in Q2 2011

Taking a closer look, the data also exhibits a strong weekly seasonality, as shown in Figure 16. This intuitively makes sense, since most peoples life follows a weekly routine. At the same time, the way this seasonality plays out can differs between regions. A region like *Midtown Center*, which is mixed residential and business, has clear rush-hour peaks and slower weekends, whereas *East Village*, known for its night clubs, has its peaks late in the evening towards the weekend. Another interesting trend can be observed at airports, where demand remains relatively constant

throughout the day.

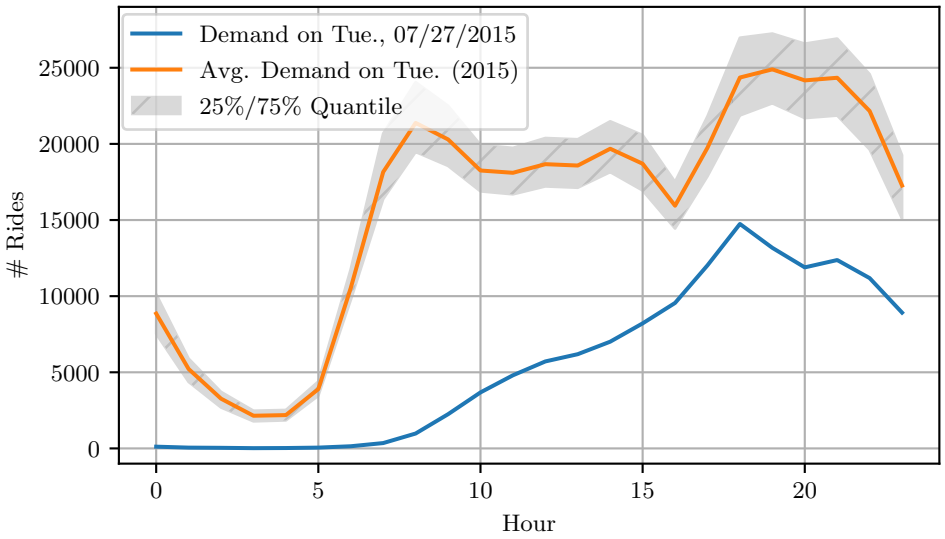

Figure 17: Taxi ridership during the blizzard of January 2015

Another source of drift in the dataset are extreme weather events, such as hurricanes or blizzards. Naturally, as the daily life of New Yorkers gets disrupted, so does the taxi demand. Figure 17 depicts the taxi demand of Tuesday, January 27th, 2015 (in blue) and the average demand on Tuesdays in 2015 (in red) as well as the 25% and 75%-quantile of the average demand. During night and early morning a Blizzard passed NYC with declared snow emergencies and a controversial subway shutdown. Such an event can be described as a sudden concept drift. Similarly, holidays or special events greatly impact citizens behaviour and can therefore be regarded sources of concept drift.

In this particular use case, it seems rather easy to identify factors (e.g. weather) which have a large influence on the prediction power of a model as well as how to include this information as predictive features. This might also be due to the nature of the overall project since nearly everyone has already used a taxi as a means of transportation. However, in hindsight it is always easy to identify unusual demand patterns and subsequently investigate the underlying reason. For a predictive model, though, this information is required in real time. In other use cases and application areas, it is generally difficult to identify influencing variables apart from the obvious dataset. In case those can be identified, it is often impossible to measure and quantify those factors, which is why they are often referred to as “hidden variables“

(Žliobaitė et al., 2016). Therefore, this work restrains the inclusion of external data sources.

In Table 3 an excerpt of the final dataframe is shown. Each column represents the demand in a single taxi zone (here the largest seven zones are shown), and each row represents one hour. Overall, the data now contains 263 columns and 83,231 for a total of almost 22 million observations.

Taxi Zone Time	79	161	162	170	230	236	237	...
2009-01-01 00:00:00-05:00	551	339	190	357	38	290	331	...
2009-01-01 01:00:00-05:00	565	220	257	472	90	273	323	...
2009-01-01 02:00:00-05:00	497	172	241	437	119	243	210	...
2009-01-01 03:00:00-05:00	563	108	238	354	231	172	133	...
2009-01-01 04:00:00-05:00	524	79	128	176	229	70	55	...
...	...	...	...	...	...	...	...	...

Table 3: Excerpt of the dataframe that is the basis for the following analysis

3.2 Predictors

It is the aim of this work to predict the taxi demand for the next hour, which can be described as a one-step out-of-sample prediction. For example, at 8am on May 20th 2019, the model is asked to predict the taxi demand for the next 60 minutes. It is assumed, that information about past rides reaches the model in real-time, therefore the entire demand history until 7:59:59am is known to the model. Looking at the data for each region separately, it can be considered an equidistant time series, as defined in Definition 1, giving the opportunity to leverage the well-established field of time series prediction. The predictors where selected to include three levels of complexity - *Baseline Predictors* that make no (or barely any) assumptions about the data; an *Established Predictor* with a strong statistical underpinning; and a *Complex Predictor* that uses a neural network. This section describes the parameters and implementations used and briefly discusses the predictive accuracy of each predictor for the year 2012, years 2009-2011 being used as training and validation data when needed.

3.2.1 Baseline Predictors

As is common practice when evaluating time series prediction, a naïve predictor is included as a baseline (Hyndman and Athanasopoulos, 2013). This predictor makes no assumptions about the data presented and uses the demand of the previous hour as the prediction for the next hour, for each region separately. Additionally, a baseline predictor that takes advantage of the weekly seasonality is used. In reference to the work of Dunning and Friedman (2017) it is referred to as the *canary* model. This predictor calculates the mean delta from the previous hour to the next hour of the past 4 weeks, and adds this delta to the observation from the last hour - again separately for all regions:

$$\begin{aligned} \hat{y}_t &= y_{t-1} + \text{mean delta of last 4 weeks} \\ &= y_{t-1} + \frac{1}{4}\sum_{i=1}^{4} y_{(t-i\times 168)} - y_{(t-i\times 168+1)} \end{aligned} \tag{3.1}$$

Given the historical data as described in Chapter 3.1, these predictions can be computed efficiently using panda's `.shift()` function[3], since the time series is equidistant (Program Code 1). Additionally, the output is clipped, since a negative demand is not possible.

Code 1 Baseline Predictors in Python with Pandas

```
1 def naive_predictions(df):
2     return df.shift(1)
3
4 def canary_predictions(df):
5     return (df.shift(1) + sum([df.shift(168*i) - df.shift(168*i + 1) \
6     for i in range(1,5)])  / 4).clip(lower=0)
```

In Figure 18 predictions for the first two days of May 2012 in the taxi zone 237, Upper East Side South, are shown. The prediction of the baseline predictor clearly lags behind the true demand by one hour, whereas the canary predictor is already capable of closely following the daily trends, in this case characterized by rush-hour peaks in the morning and evening. Though both predictions appear to be relatively

[3]Pandas is a open-source library for data analysis in Python (McKinney, 2010)

close, evaluating the error metrics for the entire year 2012 over all regions paints a clear picture. The naïve predictor has a RMSE of 49.72, whereas the canary has a RMSE of 28.97. This large difference in predictive quality was to be expected, since the canary predictor includes prior knowledge about the data (weekly seasonality) and is more robust towards outliers thanks to its moving average approach. However, in the twenty largest regions, the error significantly increases to 153.89 (naïve) and 88.19 (canary) respectively. These regions experience much higher fluctuations in taxi demand, from less than 100 overnight, to over 1500 during peak hours, whereas 171 regions never have more than 100 taxi trips in one hour. It is therefore expected to have a much higher RMSE. For the same reason, the relative error (sMAPE) is usually lower in regions with less demand (Table 4), since these models rely heavily on the single previous observation.

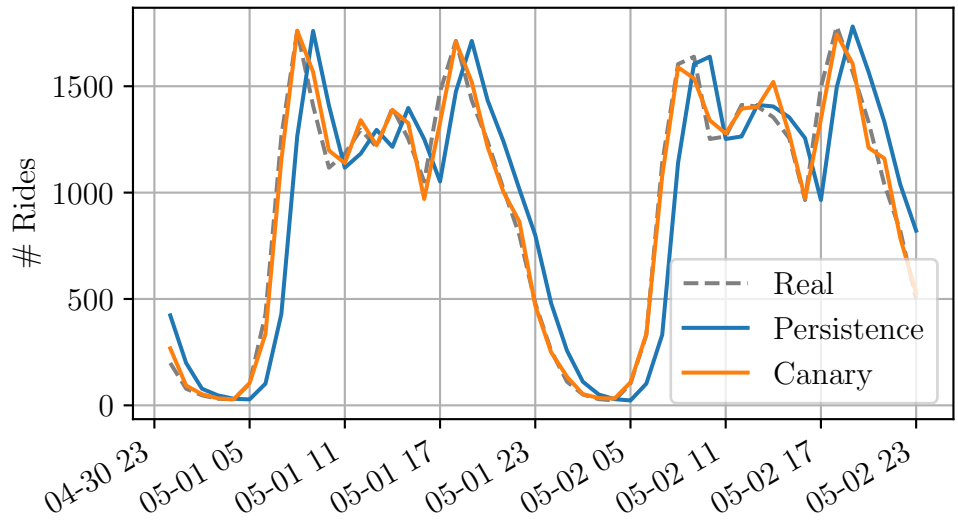

Figure 18: Predictions of the baseline predictors

3.2.2 Established Predictor

After describing and evaluating the baseline predictors, the next step is building a more capable prediction model. The recently held "M4-Competition", which is an open competition to evaluate and compare forecasting methods, has underpinned the continued importance of statistical approaches. In their conclusion paper, Makridakis et al. (2018) remark, that twelve of the seventeen most accurate models were "combinations of mostly statistical approaches". In this work, an ARIMA-model

was chosen as the statistical approach. Many related works looking to improve taxi demand prediction also include ARIMA to evaluate the accuracy of their new approach (Ke et al. (2017), Moreira-Matias et al. (2013a), Jiang and Zhang (2018)). A more in-depth explanation of the mathematical foundation of ARIMA models can be found in Chapter 2.1.3.

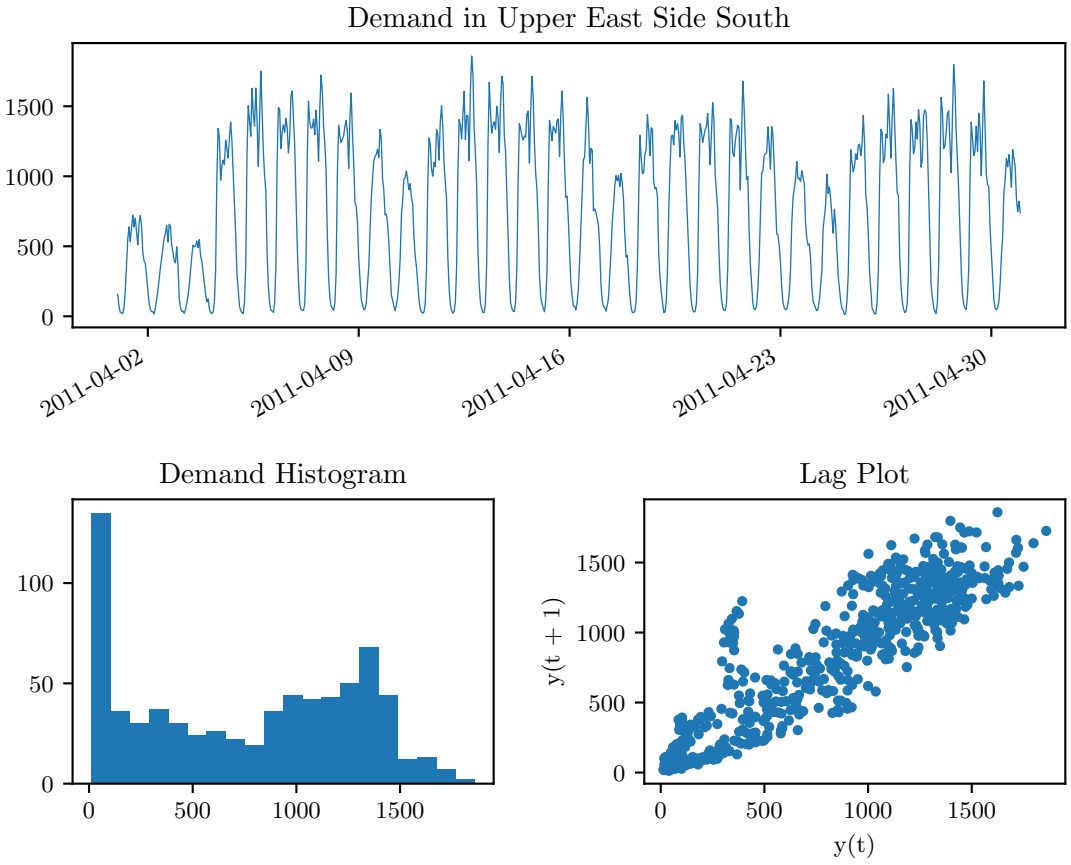

Figure 19: Visualizations to determine stationarity

In order to use an ARIMA model, the first step is to obtain a stationary series. Figure 19 shows three visualizations for taxi zone 237, "Upper East Side South", that help in determining the stationarity. The first plot shows the actual taxi demand over two weeks in 2011, where a clear seasonality is present in daily and weekly intervals. In the second plot a histogram with twenty bins is shown. For a stationary series one would expect to see a gaussian bell-like shape, which is clearly not the case. Lastly, the third plot shows the lag plot of the time series, which here appears to imply a correlation, but certainly not a clear one. It can therefore be assumed, that the time series is not stationary. Additionally to this kind of visual inspection, an Augmented Dickey-Fuller unit root test (ADF) is performed. This test evaluates the

null hypothesis of an $ARIMA(p, 1, 0)$ process against the stationary $ARIMA(p + 1, 0, 0)$ alternative (Dickey and Fuller, 1979). Using the ADF test on the same region for the year 2011 yields a test statistic of -2.3145 with a p-value of 0.1673. Therefore the null hypothesis cannot be rejected, and the time series is assumed to be non-stationary.

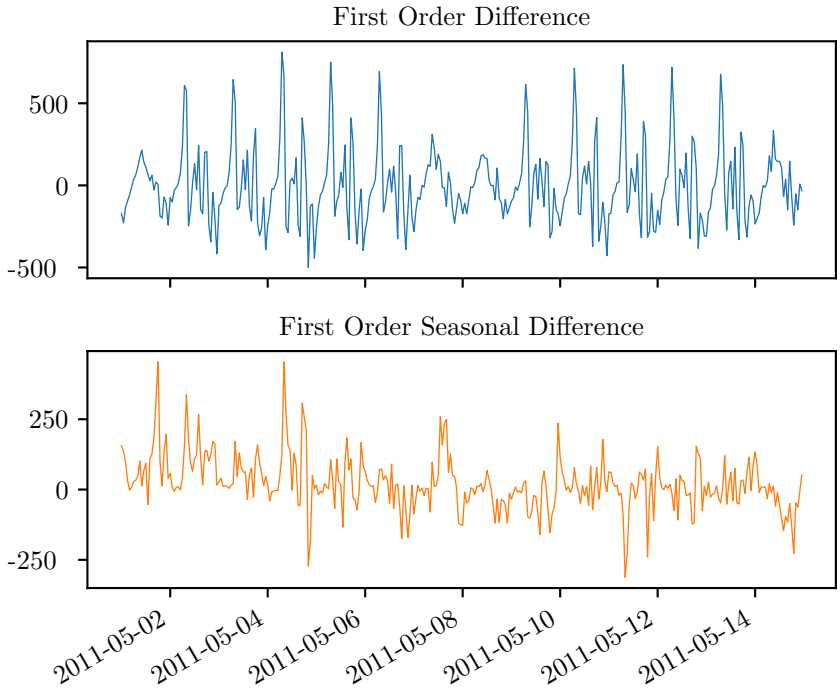

Figure 20: Differencing of the time series for the first two weeks in May

The result of taking the first order difference is shown in Figure 20 and still seems to follow a daily trend, as well as a weekly trend with lower values during weekends. As a next step, the first seasonal difference is applied using a lag of 24 for the daily seasonality. However, this still produces a time series, which seems to follow a regular pattern. Finally, seasonal differencing with a lag of 168 removes any obvious seasonality from the data. Continuing the visual evaluation, the histogram after seasonal differencing looks considerably more bell-like, and the auto-correlation plot is much clearer. Performing the ADF test leads to a rejection of the null hypothesis with a test statistic of -9.2426 and a p-value of 1.567×10^{-15}. Therefore, this differenced form of the time series can be approximated by an ARIMA model. Similarly, seasonal differencing leads to a rejection for all 263 taxi zones, with an average p-value of 1.734×10^{-16}, making it very likely that this step makes all individual time series stationary.

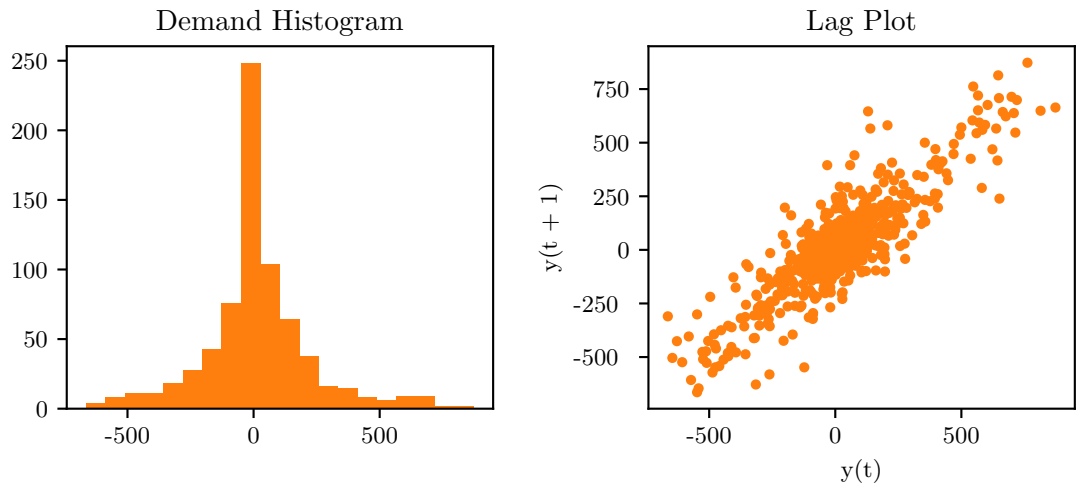

Figure 21: Determining stationarity after seasonal differencing

In order to determine the number of AR and MA terms used for the model, a first step is to look at autocorrelation and partial autocorrelation function plots, to determine a general range for the number of terms. In a next step the Akaike Information Critera (AIC) is used for tuning the hyperparameters and to determine the best ARIMA model. The AIC, published by Akaike (1998), tries to achieve a trade-off between the goodness of fit and simplicity of the model. With k being the number of parameters of the model and $\hat{L}$ the maximum value of the likelihood function, the AIC is defined as:

$$\mathrm{AIC} = 2k - 2\ln(\hat{L}) \tag{3.2}$$

Looking at the autocorrelation plot in Figure 22, the correlation is still significant with a large number of lags. However, the partial autocorrelation reveals that most of this can probably be explained by the propagation of the autocorrelation at lag 1. But since there is a strong local maximum in correlation around the 24 hour mark, the grid search for optimal hyperparameters will include p values from 20 to 28. For the number of MA terms it can be assumed they play a less significant role, and will be checked in the range from 1 to 6. At the end, an $ARIMA(24, 0, 4)$ model was chosen with an AIC of 25 798.40. It must be noted however, that the high number of observations seems to overshadow the importance of the number of parameters, and the AICs are very similar.

For this first evaluation of the predictor, the a separate ARIMA model is trained on data from 2009-2011 for each regions. Figure 23 again shows predictions for

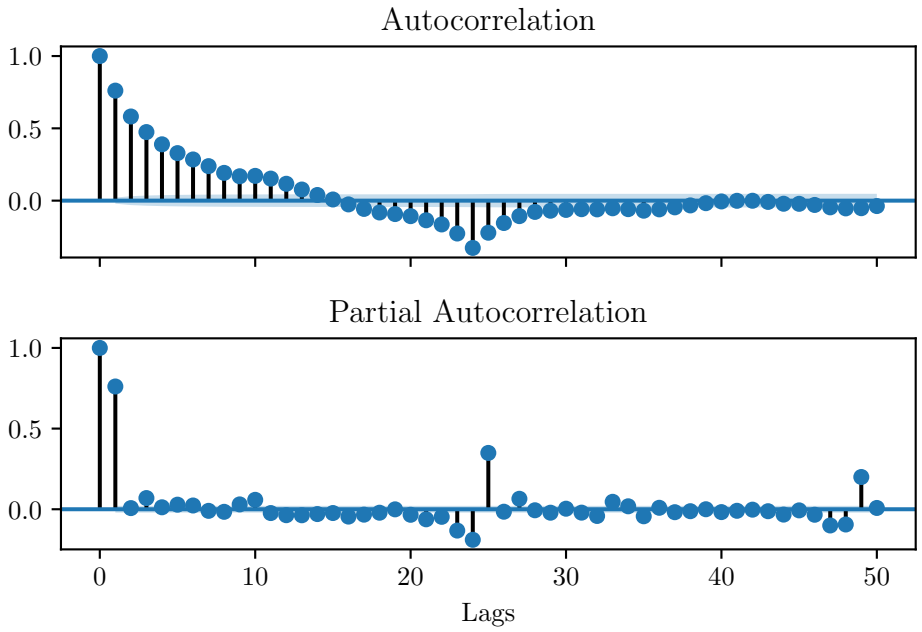

Figure 22: ACF and pACF plot for region 237

region 237, Upper East Side South, for the first two days of Max 2012. The model is clearly able to closely capture and follow the daily trends in the data. However, with a RMSE of 28.88 on all regions in 2012, it is just barely more accurate than the canary model. There are two possible explanations for this. First, as the pAFC plot (Fig. 22) showed, a lot of correlation can be explained with the first lag, which is also included in the canary model. Second, the canary model is probably more robust towards outliers, like holidays and weather events, since it uses the average of the past four weeks, whereas the ARIMA model is strongly dependent on the the previous week due the seasonal differencing.

3.2.3 Complex Predictor

As described in Section 2.1.4, a neural network autoregression model (NNAR) can be seen as an extension of the ARIMA approach without the limitations of stationarity and linearity. Therefore the first step in introducing a neural network based predictor was to use such a model. The input vector has a length of 28, containing the previous 24 observations (similar to the ARIMA model) and the 4 previous seasonal observations (similar to the canary model). In order to transform the dataset into a regression problem, each observation, meaning the amount of taxi rides during

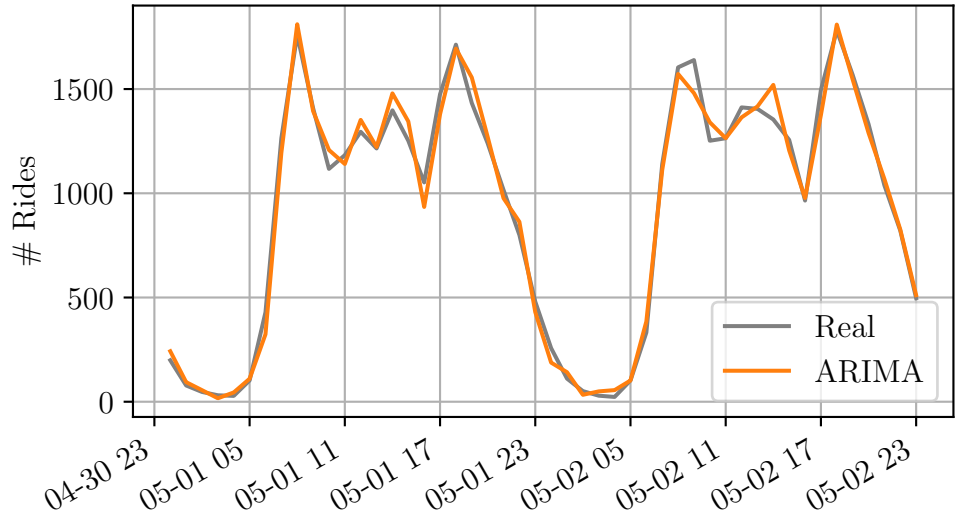

Figure 23: Predictions of the ARIMA predictor

a particular hour in a particular region is considered the label, to which a 28 dimensional input vector is created. The input observations are scaled using standard scaling, after which they have zero mean and a unit deviation. Program Code 2 shows the preprocessing steps. Note that the parameters used for scaling need to be preserved, as they are needed to make predictions later.

Code 2 Data preprocessing for the NNAR

```
from sklearn.preprocessing import StandardScaler

def generate_data(df, region, year):
    result = pd.DataFrame()
    for lag in range(1, 25):
        result['lag-\%d' \% lag] = df[region].shift(lag)[year]
    for season in range(1,5):
        result['season-\%d' \% season] = df[region].shift(season * 168)[year]
    result.dropna(inplace=True)

    scaler = StandardScaler()
    X = scaler.fit_transform(result.values)
    y = df[region][year].values
    return X, y, scaler.get_params()
```

For the neural network the implementation of the multilayer perceptron regressor from scikit-learn is used Pedregosa et al. (2011), with the default rectified linear unit

(ReLU) used as activation function and the Adam optimizer. This implementation tries to minimize the squared loss until the model stabilizes (`tol=4`). The model consists of two hidden layers with 128 and 4 neurons respectively. This model has 5,641 trainable parameters. A separate network is again trained for each region on the first three years from 2009 until 2011, leading to a sample size of 25,608 observations. Especially looking at the largest 20 regions, the trained model is able to predict the demand the most accurate so far with a RMSE of 82.56, however is does not perform much better when looking at the sMAPE. Since the model tries to minimize the squared loss it presumably fits the demand more closely during periods of high demand, where the potential penalty could be higher. During periods of lower demand, like night time, prediction errors impact the relative error sMAPE more strongly than the absolute error metric RMSE.

In a next step the NNAR model is expanded to include more information that could impact the taxi demand, and to better leverage the predictive potential of the neural network. As discussed in Section 2.1.3 the given time series is not stationary, meaning the value of the series is time-dependent. This fact can be leveraged by including time-dependent features, namely:

- **Day of the week**: The main seasonality of this series is weekly, since most people's schedule is also weekly. Using "day of the week" as a categorical feature allows the model to learn distinct differences between the days. Since the data is categorical, it is given to the model as a one-hot encoded vector with length 7.

- **Hour**: Obviously, the hour of the day impacts the demand very strongly. The hour of the day is cyclic in nature and should therefore not be used directly, as the model would have no idea, that 23 immediately precedes 0. Therefore the sine and cosine with a period of $\frac{24}{2\pi}$ is applied, to map the values hour onto points on the unit circle.

- **Month** Similarly, the month of the year is a cyclical feature and transformed using sine and cosine, with a period of $\frac{12}{2\pi}$. By including the month the model might be able to learn some trends with yearly seasonality.

Lastly, instead of treating the regions separately a single model is trained. To

achieve this another categorical feature is included to denote the region a taxi ride has started. Since there are 263 taxi zones in NYC, this feature would dramatically increase the complexity of the model and the number of parameters to be estimated. The scope of the complex model is therefore limited to the twenty busiest regions (listed in Appendix A.1), which make up about 60% of the total demand. Overall this final model therefore has a 59-dimensional input vector. For predictions made in 2012 this complex model performs the best out of the five tested models with a RMSE of 60.98 and a sMAPE of 5.46%.

3.2.4 Comparison of Predictors

Predictor	RMSE (all)	RMSE (Top 20)	sMAPE (all)	sMAPE (Top 20)
Naïve	49.72	153.89	23.16%	13.66%
Canary	28.97	88.19	24.29%	7.59%
ARIMA	28.88	88.00	24.01%	7.91%
NNAR	27.22	82.56	24.51%	7.38%
Complex	–	60.98	–	5.46%
Mean	106.53	358.27	63.89%	27.21%

Table 4: Evaluation of predictors in 2012

The evaluation of the models is summarized in Table 4. It also includes the predictor "Mean", whose prediction is always the mean demand of the region in 2011. The fact that the overall demand of the regions has a tremendous range from 22 to 6.3 million in 2012 heavily impacts the statements that can be derived from the error metrics. All models have a lower squared error on all regions compared to the Top 20, and at the same time all models have a higher relative error on all regions. Figure 24 illustrates the large differences in demand, where the region with the 50th highest demand, Alphabet City, has a demand that is more than 8 times lower, than the mean demand in the Top 20 regions (573 rides/hour vs. 69 rides/hour). A low mean demand also implies a low amount of fluctuation in hourly rides. This is the main factor, why predictions over all regions appear to be more accurate in terms of the squared error, exemplified by the *Mean* model. On the other hand, the different orders of magnitude cause the large difference in the relative error,

where underestimating the mean by e.g. 30 rides leads to a sMAPE of around 30% in Alphabet City, the same (absolute) error in the Upper East Side South yields a sMAPE of less than 3%.

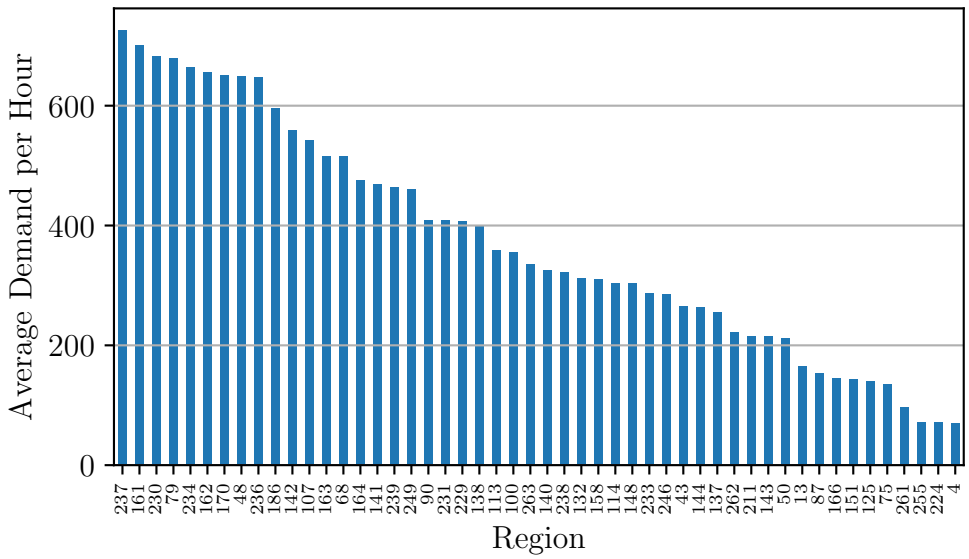

Figure 24: Average hourly demand in 2012 in the 50 busiest regions

3.3 Drift Detectors

After implementing a set of predictors, the next step is to look at drift detectors. So far the scope has been purposefully limited to 2012. However, when increasing the forecast horizon the accuracy will presumably decrease, as the concepts learned from 2009 to 2011 do not apply fully anymore. The presence of this effect will be studied closely in Section 4.1. In order to deal with concept drifts efficiently, drift detectors are needed, that indicate, when a retraining or model switch is necessary. First, the implementations of the drift detectors from Section 2.2.2 are shown along with details on their integration into the taxi demand task. Then, strategies for detecting sudden concept drift, including the novel error intersection approach are presented.

3.3.1 Detecting Incremental Concept Drift

As per the first research question "How can we address concept drift in regression problems in a real-world context?", two different approaches will be evaluated – implicit drift detection by periodically retraining, and explicit drift detection, which triggers a retraining.

Code 3 Python pseudo-code for implicit drift detection

```
1 model = Model(history, learn_range='3 years')
2
3 for data in get_new_data():
4         model.update(data)
5         prediction = model.predict()
6
7         if model.get_age() > MAX_AGE:
8                 model = Model(model.history, learn_range='3 years')
9         yield prediction
```

An implicit approach to drift detection only consists of a model, and some logic to trigger a retraining. Code 3 illustrates this approach using python-like pseudo-code. Once the model is older than a pre-defined threshold, it is replaced by a new model, that has been trained on the most recent 3 years of data (line 8). This sliding window approach balances the need of the model to see multiple seasons in order to learn the concepts effectively with the need to focus on newer data, to learn the most recent concepts. In Section 4.2 this approach is evaluated using thresholds of one year, six months, and three months.

3.3.2 Detecting Sudden Concept Drift

As shown in Section 3.1, a potential source of sudden concept drift can be extreme weather events. The models are purposefully not aware of weather conditions when making their predictions, therefore weather can be considered a hidden variable. As part of the second research question "How can we leverage models with different degrees of complexity to make the prediction more robust?", multiple approaches were designed, that leverage models of different complexity.

Ensemble methods have been widely used in concept drift adaptation methods (Gama et al., 2014). Usually, these approaches rely on the combination of various models with an average of the delivered predictions to increase the overall performance. However, in this work, the evaluate is focused on approaches which use the predictions issued by two static models of different complexity to detect drift in the data and adapt the prediction accordingly. Different complexity here means that a simple model (M_{simple}), which is strongly influenced by the most recent demand, is combined with a more sophisticated, learned model ($M_{complex}$). In general, the ensemble should gravitate towards the predictions from $M_{complex}$, because it successfully captures the general demand structure and is therefore able to compute accurate predictions for the taxi demand in the respective taxi zone. However, during times with sudden concept drifts, $M_{complex}$ cannot provide accurate predictions since the demand patterns clearly deviate from the usual trajectories. In these cases, the ensemble should prefer M_{simple} because it can quickly adapt to current changes in demand.

For the evaluation, a number of possible drift detectors are examined. As M_{simple} and $M_{complex}$ the Naïve and NNAR models will be used respectively. The decision, whether a model switch is performed, can be based on one of the following approaches:

- **Drift Detector Approach (DDA)**: Using a detector, such as ADWIN or EDDM a switch will be performed, when a drift is detected, instead of retraining the model.
- **Error Intersection Approach (EIA)**: Determine model to be used based on the EWMA of the prediction errors in the last 6 hours.
- **Biased (Error) Overlap Approach (BOA)**: Use the prediction of the simple model only if its prediction was better in the last three consecutive hours.

Usually, the detectors like EDDM expect an input of *TRUE* or *FALSE*, depending on if the model has made a correct classification. For the regression task at hand, a prediction is considered to be correct, if one of two conditions is met:

i) $\frac{|\hat{y}-y|}{y} \leq 20\%$, i.e. the error is less than 20%

ii) $\hat{y} \leq 100 \wedge y \leq 100$, i.e. both predicted and real demand are below 100

The choice of the relative threshold and absolute cut-off is influenced both by the accuracy of the model and the business requirements. The cut-off at 100 is meant to mitigate the fact, that the relative error is more sensitive in times of low demand, e.g. over night. About 10% of all observations in the Top 20 regions are lower than 100. A visualization of how this classification of the predictions plays out for the NNAR model is shown in Figure 25.

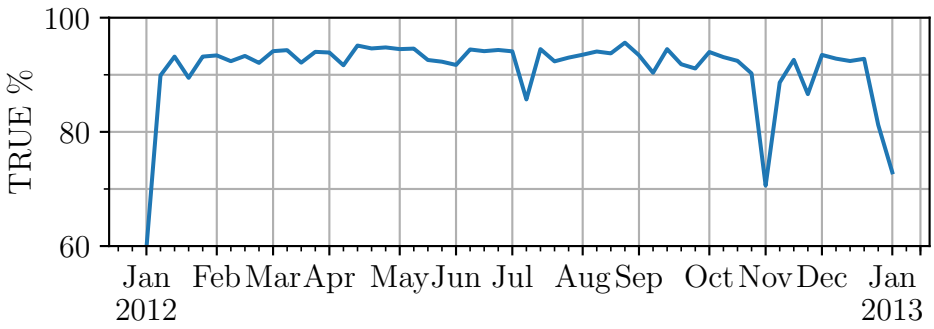

Figure 25: Weekly percentage of correct predictions from the NNAR model

Code 4 Outline of explicit drift detection in python pseudo-code

```
current_model = ComplexModel()
other_model = SimpleModel()
detector = Detector()

for data in get_new_data():
        current_model.update(data)
        other_model.update(data)
        prediction, last_prediction = current_model.predict(), prediction
        correct = (last_prediction <= 100 and data <= 100) \
                or abs(prediction - data)/data <= 0.2
        detector.add_element(correct)
        if detector.detected_change():
                current_model, other_model = other_model, current_model
                detector.reset()

        yield prediction
```

The python library *scikit-multiflow* was used (Montiel et al., 2018) for its drift detector implementations. Scikit-multiflow is a machine learning library inspired by the

MOA library[4] for multi-output/multi-label and stream data. The detectors from the library are updated by calling the `add_element()` function of the detector instance. `detected_change()` returns *TRUE* if the drift threshold has been surpassed. The novel approaches introduced in this work (EIA, BOA) follow the same interface, but operate on the classification error directly. The warning level that models like DDM support will not be used. An outline of how the detector is integrated in the prediction process is shown in Code 4.

Lastly, these approaches are compared to a more traditional ensemble set-up, where both models are used concurrently, since existing drift handling strategies for regression are usually built this way (see Section 2.2.3). We compute the exponential weighted moving average (EWMA) of the last 6 prediction errors of both models respectively and determine the sum of errors. Subsequently, the contribution of each model is computed as the sum of errors to determine the weights of each model for the ensemble prediction (e.g. if $M_{complex}$ accounts for $\frac{1}{3}$ of the sum of errors, its weight for the next prediction is $\frac{2}{3}$).

[4]https://moa.cms.waikato.ac.nz/

4 Evaluation

So far, the models presented in Section 3.2 have been limited to data from 2009 to 2012, using the first three years for training and the last year for a baseline evaluation. Through this self-imposed limitation it is ensured that the model development and the choice of hyperparameters happen in isolation from any concept drifts that occur over time. In a way, the models were developed *as if* it was the beginning of 2013. In this chapter, we fast-forward to 2018 and evaluate the model performance over the next six years.

The evaluation of this work is split into three sections. First, the models from Section 3.2 are evaluated over the remaining six years of the dataset to determine if a decrease in accuracy takes place, indicating a change in concepts. Next, established strategies that could be applied to the problem are compared based on their accuracy. Lastly, multiple strategies for combining a static and a dynamic model are tested, including the novel error interception approach.

4.1 Presence of Concept Drift

In order to determine the actual presence of concept drift in the taxi demand of New York City, the predictions of the models from Section 3.2 over the entire available time range is evaluated. Under the assumption that no concept drift is present, the accuracy is expected to remain constant or improve over time (see Section 2.2.1). And in fact, comparing the RMSE of each month from 2012 to 2018 (Figure 26) seems to hint at a reduction in errors, and therefore an increase in accuracy. However, the observation that especially the naïve prediction improves dramatically indicates that another factor is at play. Comparing the progression of the naïve prediction error with the overall demand (Figure 15) shows striking similarities.

To quantify the correlation between the total number of rides and the prediction error, the Pearson correlation coefficient can be used. The Pearson correlation coefficient is a measure of linear correlation between two variables with a range from -1 (perfect negative correlation) over 0 (no correlation) to $+1$ (perfect positive

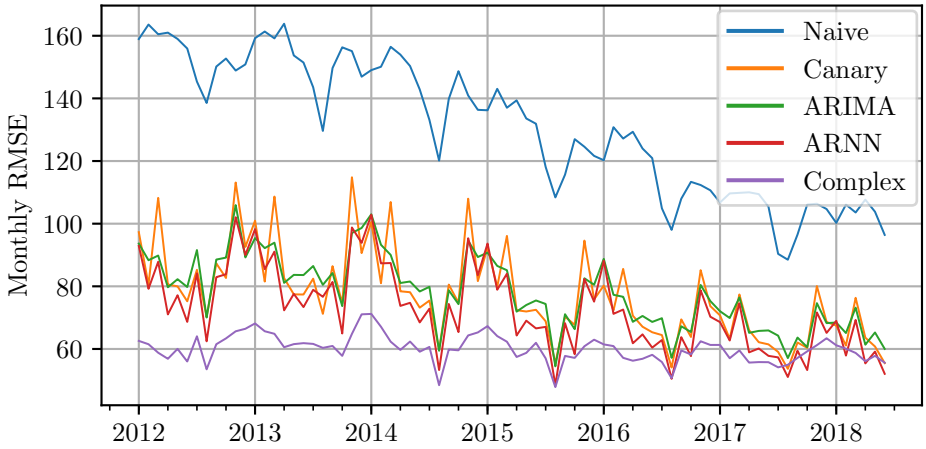

Figure 26: Monthly RMSE of all predictors

correlation). It is defined using the covariance $\text{cov}(x, y)$ and variance $\text{var}(x)$ as:

$$r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)} \cdot \sqrt{\text{var}(y)}} \tag{4.1}$$

The Pearson correlation coefficient between the monthly RMSE of the naïve prediction and the monthly taxi demand is 0.987, indicating a strong positive correlation, confirming the suspicion from earlier. Therefore, the increase in accuracy can be almost entirely explained by the overall decrease in taxi demand. Similarly, the correlation can be considered strong for all models except for Complex (see Table 5). Since the overall decrease in demand is so high (173.8 million in 2012; 111.7 million in 2017), the squared error by itself cannot be relied upon to evaluate the accuracy over time.

Model	ARIMA	NNAR	Canary	Complex	Naïve
RMSE	0.717950	0.619591	0.645408	0.433092	0.987329
sMAPE	-0.440448	-0.325020	-0.215905	-0.855102	-0.147284

Table 5: Pearson correlation coefficients between monthly errors and overall demand

Instead, the sMAPE can give a better overall impression on the accuracy evolution over time, since the correlation is (for the most part) much weaker. Figure 27 shows how the monthly sMAPE develops over time. The naïve baseline model remains relatively constant, while all other models decrease in accuracy. Especially the complex model is affected, increasing the yearly sMAPE from 5.46% to 8.57%.

Interestingly, the sMAPE of the ARIMA model remains fairly constant over the first four years, only to then increase rapidly from 8.17% in 2015 to 9.45% in 2018. It can therefore be concluded that an incremental concept drift is taking place, affecting the accuracy of all models, but especially those with increasing complexity.

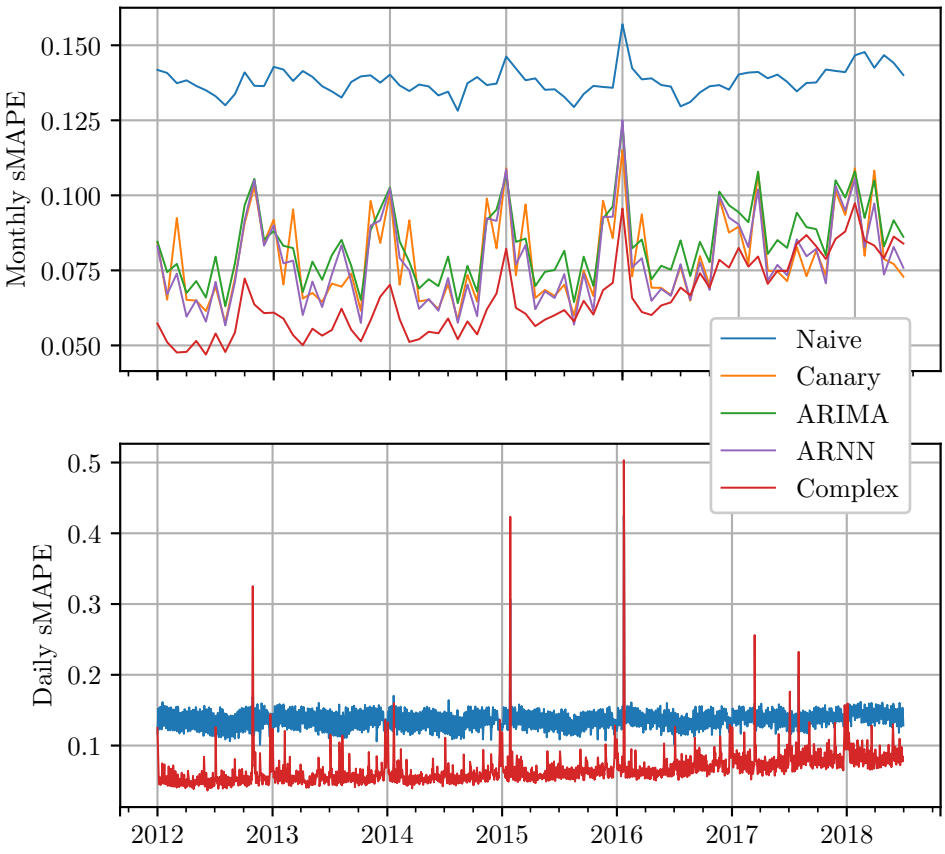

Figure 27: Monthly and daily sMAPE of all predictors

Looking at the sMAPE with a daily resolution, another pattern emerges. While the error of the naïve predictor remains fairly constant, the error of the complex model contains many spikes, of which six clearly surpass the naïve prediction. The spike at the end of 2012 for example can be explained by Hurricane Sandy, which reached New York on October 29th; the largest spike at the beginning of 2016 coincides with a blizzard hitting the city. These short periods can be considered a sudden concept drift instead of outliers since they are not a "once-off random deviation or anomaly" (Gama et al., 2014), but instead can be explained in hindsight.

As noted by Tsymbal et al. (2008), a common real-world case is that concept drift occurs only locally within the dataset. Given the natural segmentation of the taxi demand into regions, a next step is to evaluate regional differences in the amount

of concept drift. Figure 28 shows the increase in sMAPE over time, averaged for the three regions with the highest and lowest absolute change from 2012 to 2018. Within this dataset it can be assumed, that all regions show some form of concept drift, but it is important to note, that there are considerable differences in severity, even within only the Top 20 zones.

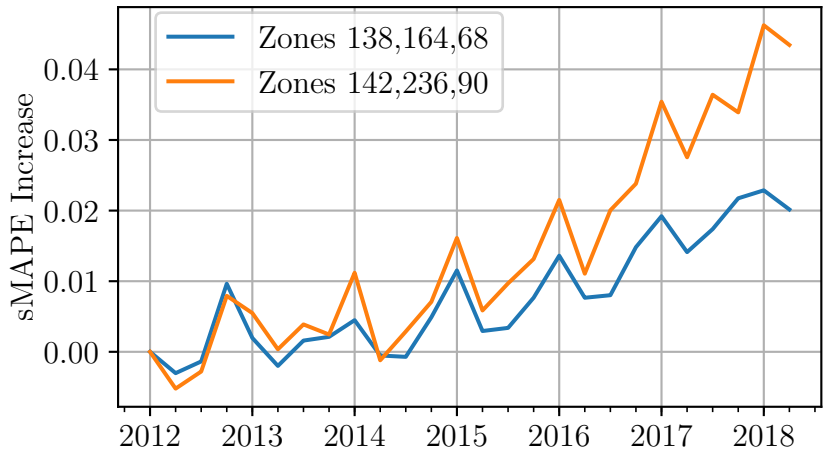

Figure 28: Quarterly sMAPE of the zones with the highest and lowest change in accuracy

In this first section of the evaluation, three key conclusions were drawn: First, because the strong downward trend of overall taxi demand, the sMAPE needs to be considered when comparing the accuracy of different time frames. Second, the sMAPE of all models increases over time except for the naïve model, indicating that concept drift is in fact present. Measures to remain accurate predictions are evaluated in the next section. Third, short spikes in the mean error can be considered sudden concept drift, often caused by unusual weather or holidays. The novel error intersection approach that counteracts this effect is evaluated in Section 4.3

4.2 Handling of Incremental Concept Drift

In this section, implicit approaches to handle incremental concept drift are evaluated. As discussed in Section 2.2.1, an implicit approach does not actually determine whether a shift has occurred, but instead regularly updates the model independently. Therefore it might happen, that the retraining did not actually improve the performance, and was basically a waste of time. Additionally, depending on the learning

algorithm, the new model might behave differently in certain edge-cases, and the experience from using the old model cannot help anymore (Dunning and Friedman, 2017). Nonetheless, evaluating the implicit approach can help to determine, whether the increase in error is caused by a shift in the dataset, which the model can capture and learn, or if there are unknown variables at play, that the model does not know about.

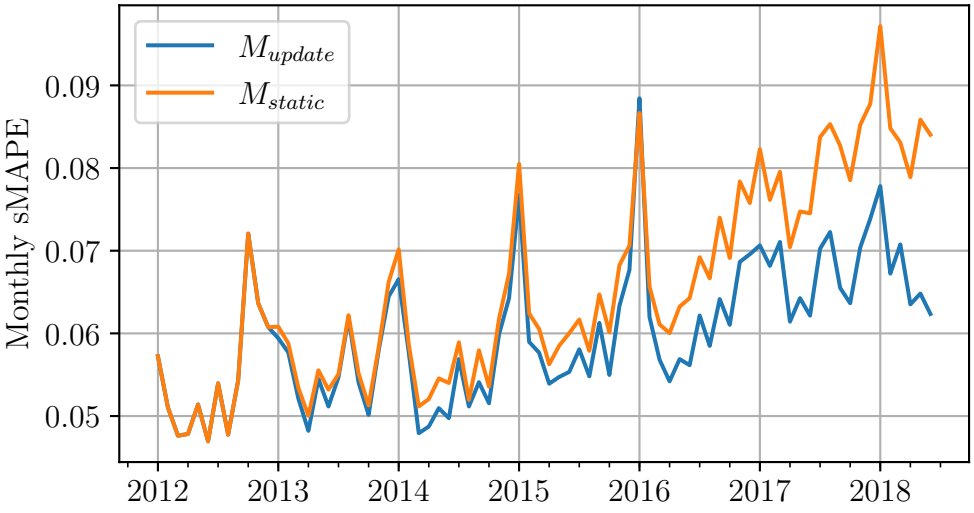

Figure 29: Monthly sMAPE comparing M_{update} with M_{static}

For this evaluation the model M_{update} is updated yearly, and trained each time using the previous 3 years of data. This means that the forecast for 2013 is performed with a model trained on data from 2010-2012, the forecast for 2014 is issued by a model trained on data from 2011-2013. Therefore both approaches had an equal amount of data available for training. The results are shown in Figure 29 and show that M_{update} has a lower error in every month after 2012, except for January 2016, where a blizzard hit the city that left a city-record 70cm of snowfall, impacting the daily life for multiple days. This sudden concept drift was not captured by either model, and impacted the mean error for that month.

In Table 6 the yearly errors are compared. Similarly to the percentage error, the squared error improves every year, with the gap increasing every year. Additionally, the p-value of the two-sided Diebold-Mariano Test (DM-Test) is shown. For every year the test has a very low p-value, leading to a rejection of the null hypothesis and the conclusion that the forecasts actually have a statistically significant difference in accuracy. Overall, the sMAPE was improved from 6.54% to 6.00%, and the RMSE

from 54.38 to 50.48. Since predictions on the same regions over the same time period are evaluated, both metrics can be used.

	sMAPE		RMSE		
Year	M_{static}	M_{update}	M_{static}	M_{update}	p-value (DM-Test)
2012	5.46%	5.46%	54.54	54.54	1.64×10^{-1}
2013	5.67%	5.56%	56.61	55.06	7.29×10^{-23}
2014	5.77%	5.50%	55.57	52.93	5.55×10^{-31}
2015	6.35%	6.00%	54.19	51.14	2.42×10^{-39}
2016	6.95%	6.38%	53.21	48.04	3.86×10^{-45}
2017	8.01%	6.80%	52.88	44.87	9.95×10^{-118}
2018	8.57%	6.81%	52.90	42.99	7.04×10^{-58}
Overall	6.54%	6.00%	54.38	50.48	9.15×10^{-294}

Table 6: Yearly evaluation of M_{update} and M_{static}

Furthermore, different update intervals were examined. Table 7 compares the prediction errors of three intervals: *Yearly* - updating on January 1st, *Half-yearly* - additionally updating on July 1st, and *Quarterly* - additionally updating on April 1st and October 1st. For each update the model is retrained using the most recent three years of data at that time. In general the evaluation indicates, that updating too frequently can also have a negative impact, as the quarterly updated model performs consistently worse. It seems like choosing a biyearly update strategy achieves a balance between drift adaptation and unnecessary computations.

In conclusion the data shows, that a time-triggered retraining of the model does improve its predictive performance in a way, that is statistically significant. Given the consistency of improvement, retraining and model replacement could be automated and should not need any further manual input.

4.3 Ensemble Methods for Concept Drift

In this section, the strategies for handling sudden concept drift as introduced in Section 3.3.2 are evaluated. Table 8 shows the results for the overall prediction performance of the different approaches on the dataset from 2012 to 2018. The

	sMAPE			RMSE		
Year	Quarterly	Half-yearly	Yearly	Quarterly	Half-yearly	Yearly
2012	5.45%	5.48%	5.46%	54.28	54.39	54.54
2013	5.60%	5.60%	5.56%	55.19	54.99	55.06
2014	5.59%	5.53%	5.50%	53.20	52.77	52.93
2015	6.09%	5.98%	6.00%	50.69	50.54	51.14
2016	6.54%	6.34%	6.38%	47.82	47.61	48.04
2017	6.70%	6.73%	6.80%	44.45	44.28	44.87
2018	7.23%	6.81%	6.81%	44.84	42.99	42.99
Overall	6.02%	6.01%	6.00%	50.92	50.17	50.48

Table 7: Evaluating different update intervals

first part shows the baseline approaches M_{simple}, $M_{complex}$ and *Ensemble*. M_{simple} is the naïve predictor introduced in Section 3.2.1, $M_{complex}$ the expanded NNAR from Section 3.2.3, and *Ensemble* calculates a weighted sum of both models based on the exponentially-weighted error of a sliding six-hour window. This ensemble does not perform better than $M_{complex}$ does on its own. Apparently the predictions from M_{simple} are so far off, that they should not be included constantly. The second part of the table shows the drift detector approaches (DDA), where the model switch decision is based on a standard drift detector (see Section 3.3.2). Their accuracy is much more similar to M_{simple}, indicating that it was active for most of the predictions. Closer analysis showed, that the detectors were much less likely to be triggered when the simple model was active, since its error does not fluctuate as strongly (see Figure 27). Finally, both BOA (Biased Error Overlap Approach) and EIA (Error Intersection Approach) are able to improve on the accuracy of $M_{complex}$. The absolute difference in RMSE between them and $M_{complex}$ appears to be rather small. However, it has to be considered that over 1.3 million predictions are made. Using the DM-Test between the predictions of BOA and $M_{complex}$ yields a p-value of 1.89×10^{-7}, therefore the difference is statistically significant. BOA and EIA on the other hand, do not differ significantly (p-value: 0.183). For simplicity, further analysis will be focused on EIA.

The effectiveness of *EIA* is illustrated in Figure 30, which depicts the demand predic-

Approach	# Switches	sMAPE	RMSE
M_{simple}	n/a	13.80%	115.871
$M_{complex}$	n/a	6.00%	50.478
Ensemble	n/a	6.75%	58.381
DDA_{PH}	1,811	9.79%	82.176
DDA_{ADWIN}	70	11.64%	97.657
DDA_{EDDM}	30	13.47%	112.783
BOA	397	5.99%	50.391
EIA	365	5.98%	50.370

Table 8: Comparison of sudden drift detectors

tions during the blizzard in 2015 (see Figure 17), as an example for sudden concept drift. At the beginning, *EIA* (red dashed line) always chooses $M_{complex}$ (blue line), hence they are right on top of each other. However, at 4pm on January 26th (marked by a black vertical line), the approach switches to M_{simple} (orange line), which can quickly adapt to the unusual demand pattern. $M_{complex}$ clearly fails to predict this behavior correctly (e.g., peak at around 5am on 01-27).

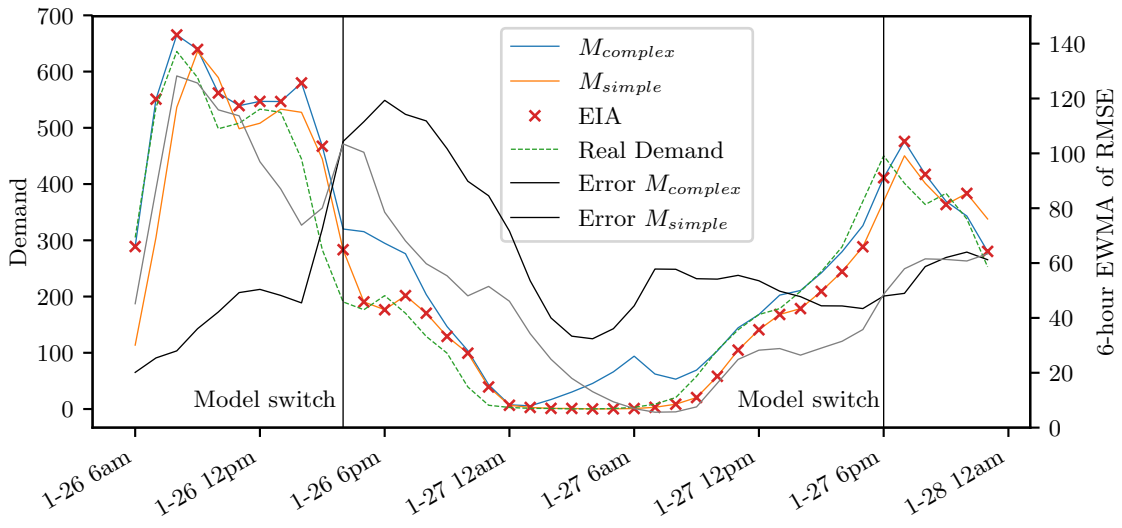

Figure 30: Predictions of EIA during the blizzard on 2015-01-27

Additionally, analyzing the predictive performance in those hours where M_{simple} is chosen by *EIA* (706 out of 56,951 hourly forecasts in total) shows that the approach improves prediction performance on average by 8.4% (RMSE EIA: 75.31 vs RMSE $M_{complex}$: 82.21). Next, the behaviour of *EIA* is examined at a daily resolution. We analyze for which days M_{simple} is applied the most. This way, the days with most significant concept drifts can be identified in hindsight. Table 9 shows an excerpt of days with a frequent use of M_{simple} for a prediction as well as the corresponding special events (*drift causes*) on that day. In most cases, drift is triggered by unusual weather events or public holidays. The third column depicts the relative improvement in RMSE of *EIA* compared to predictions by $M_{complex}$ alone. The full table can be found in Appendix A.2. While the approach improves the prediction overall, there are exceptions. For example, on Christmas Day 2014 71% of the predictions came from M_{simple}, but the error on that day was actually increased by 10% compared to $M_{complex}$ alone.

Date	Predictions by M_{simple}	RMSE Improvement	Drift Cause
2012-07-04	58.33%	7.57%	4th of July
2012-10-29	91.67%	31.72%	Hurricane Sandy
2012-10-30	91.67%	21.89%	Hurricane Sandy
2012-12-25	70.83%	5.66%	Christmas Day
...	...	...	...
2014-12-25	71.04%	-10.25%	Christmas Day
...	...	...	...

Table 9: Excerpt of days with frequent use of M_{simple} for a prediction

Lastly, the effect *EIA* has on each region is examined. Table 10 depicts these effects. In 13 of the 20 regions the RMSE was improved, 9 of which were improved significantly, according to the DM-Test. Only in one region, Upper East Side South, the approach actually worsened the prediction significantly, changing the RMSE from 61.11 to 61.24. Overall, the analysis emphasizes the importance of evaluating concept drift in sub-sections of the data, as concept drift might occur only locally.

In conclusion, the proposed error intersection approach is capable of detecting sudden concept drifts and switch the models accordingly. Most of the time, in most

	Significant	Not Significant
Improvement	48, 68, 79, 90, 138, 163, 164, 186, 249	107, 161, 162, 170
No Improvement	237	141, 142, 230, 234, 236, 239

Table 10: Significance of Improvements for each Region with 95% Confidence

of the regions, the predictive performance can be improved. While, there are some small exceptions, the overall accuracy is improved significantly.

5 Conclusion

In this thesis, the effects of concept drift on supervised regression tasks have been analyzed. Drift handling strategies for both sudden and incremental concept drift were presented and evaluated on the example of taxi demand in New York City, while introducing the novel "error intersection approach" (EIA).

The NYC taxi dataset is especially suited for answering the first research question, "How can we address concept drift in regression problems in a real-world context?". The assumption was made, that there might be two forms of concept drift in the dataset. First, long-term change due to overall shifts in the city dynamics. Second, sudden drift during special events or unusual weather occurrences. For this thesis the task was to predict the demand in each region for the next hour. Analyzing the history and current state of research in the domain of time series prediction lead to the development of multiple candidate models, with differing degrees of complexity.

The evaluation of these models showed, that all of them predicted significantly worse over time, indicating that concept drift is in fact present in the dataset. While analyzing the long-time trend, interpretation of the error metrics proved to be surprisingly challenging. The distribution and overall level of demand has to be taken into account, especially considering the 35% decrease in yellow taxi ridership from 2012 to 2017 and the imbalanced distribution of demand over taxi zones. To adequately capture the development in accuracy, both relative and absolute measures have to be taken into account.

In order to mitigate the effects of long-term incremental concept drift, multiple update intervals for implicit drift handling were tested. The comparison showed that more frequent model updates do not guarantee an improvement, as the quarterly updating model performed worst. Further analysis can be done to determine ideal retraining intervals to reduce the amount of unnecessary computations.

The second research question, "How can we leverage models with different degrees of complexity to make the prediction more robust?", lead to the development of EIA. This approach, in its core, depicts a strategy to switch between the application of simple and complex prediction models which is designed to deliver superior performance results in real-world data sets prone to concepts drifts. The hypothesis

being, that the drift detector can leverage the individual strengths of each model, switching to the simpler model if a sudden drift occurs and switching back to the complex model for typical situations. On the NYC taxi dataset many events can be classified as concept drift, since the models tested are not aware of e.g. holidays or extreme weather events like blizzards. The EIA was compared against typical predictive models for regression tasks and it was shown to outperform all regarded baselines significantly.

However, the results presented in this thesis have limitations that require further research. Compared to state-of-the-art approaches in the domain of taxi demand prediction, the predictors in the EIA-ensemble are not very complex. Deep learning approaches, like FCL-Net or ST-ResNet, might be less affected by concept drift, as they learn more complex relationships between the taxi zones. But as these models are only evaluated on shorter test periods (2 and 6 months respectively), a comprehensive long-term evaluation of these models is needed. Still, the main limitation of the EIA approach is that true value for a delivered prediction need to be acquired afterwards, which might not be the possible in all applications. However, this limitation also holds true for established drift detection methods.

The work in this thesis underlines the benefits of ensemble-based approaches when working with real-world data. In an architecture like the Rendezvous Architecture, loosely coupled ensembles fit right in and can be incorporated easily. While the core idea of switching between models falls in line with the notion of a rendezvous server, more investigations are necessary to determine the modifications needed to also enable retraining. In future work the complexity of EIA itself could be increased, by either combining multiple drift detectors or by leveraging more than two models.

A Appendix

A.1 List of Twenty Busiest Taxi Zones

LocationID	Borough	Zone	Total # of Rides
48	Manhattan	Clinton East	45.759.269
68	Manhattan	East Chelsea	36.410.981
79	Manhattan	East Village	48.118.532
90	Manhattan	Flatiron	28.982.747
107	Manhattan	Gramercy	38.367.834
138	Queens	LaGuardia Airport	31.033.642
141	Manhattan	Lenox Hill West	34.172.089
142	Manhattan	Lincoln Square East	41.102.133
161	Manhattan	Midtown Center	51.588.779
162	Manhattan	Midtown East	48.032.978
163	Manhattan	Midtown North	38.123.963
164	Manhattan	Midtown South	34.286.742
170	Manhattan	Murray Hill	47.204.649
186	Manhattan	Penn Station/Madison Sq West	44.838.684
230	Manhattan	Times Sq/Theatre District	49.046.576
234	Manhattan	Union Sq	47.167.751
236	Manhattan	Upper East Side North	48.057.999
237	Manhattan	Upper East Side South	53.724.226
239	Manhattan	Upper West Side South	34.657.551
249	Manhattan	West Village	32.576.028

A.2 List of Days, Where M_{simple} Is Used Most Often

Date	Predictions by M_{simple}	RMSE Improvement	Drift Cause
2012-07-04	58.33%	7.57%	4th of July
2012-10-29	91.67%	31.72%	Hurricane Sandy
2012-10-30	91.67%	21.89%	Hurricane Sandy
2012-12-25	70.83%	5.66%	Christmas Day
2013-01-01	62.50%	7.75%	New Years Eve
2013-07-04	58.33%	0.43%	4th of July
2013-12-25	46.04%	-16.28%	Christmas Day
2014-01-22	45.83%	-21.59%	Snow Storm
2014-12-25	71.04%	-10.25%	Christmas Day
2015-01-27	83.33%	13.94%	Blizzard
2015-12-25	66.66%	-0.75%	Christmas Day
2016-01-23	66.66%	37.52%	Blizzard
2016-01-24	100.00%	20.20%	Blizzard
2016-01-25	45.83%	-1.44%	Blizzard
2016-07-04	50.20%	-2.74%	4th of July
2016-12-25	62.50%	-1.20%	Christmas Day
2016-12-26	62.50%	0.76%	Christmas
2017-03-14	71.04%	48.46%	Blizzard
2017-07-04	87.50%	23.43%	4th of July
2017-12-25	58.33%	-2.97%	Christmas Day
2018-01-04	62.50%	6.61%	Blizzard
2018-03-21	58.54%	7.17%	Blizzard

References

Abduljabbar, R., Dia, H., Liyanage, S., and Bagloee, S. A. (2019). Applications of artificial intelligence in transport: An overview. *Sustainability (Switzerland)*, 11(1).

Akaike, H. (1998). Information Theory and an Extension of the Maximum Likelihood Principle. In *Nurs Times*, pages 199–213. Springer US.

Baena-García, M., Campo-Ávila, J., Fidalgo-Merino, R., Bifet, A., Gavald, R., and Bueno, R. (2006). Early Drift Detection Method. *Fourth international workshop on knowledge discovery from data streams*, 6:77–86.

Baier, L., Kühl, N., and Satzger, G. (2019). How to Cope with Change? Preserving Validity of Predictive Services over Time. In *Hawaii International Conference on System Sciences (HICSS-52)*, pages 1085–1094.

Bifet, A. and Gavaldà, R. (2007). Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448, Philadelphia, PA. Society for Industrial and Applied Mathematics.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (1970). *Time series analysis: forecasting and control*. John Wiley & Sons.

Brockwell, P. J. and Davis, R. A. (2002). *Introduction to Time Series and Forecasting*. Number 22 in Springer Texts in Statistics. Springer New York, New York, NY, 2 edition.

Brown, R. G. (1959). *Statistical forecasting for inventory control*. McGraw-Hill, New York, NY.

Cavalcante, R. C., Minku, L. L., and Oliveira, A. L. (2016). FEDD: Feature Extraction for Explicit Concept Drift Detection in time series. *Proceedings of the International Joint Conference on Neural Networks*, 2016-Octob:740–747.

Cavalcante, R. C. and Oliveira, A. L. (2015). An approach to handle concept drift in financial time series based on Extreme Learning Machines and explicit Drift Detection. *Proceedings of the International Joint Conference on Neural Networks*, 2015-Septe.

Chen, Chiang, and Storey (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, 36(4):1165.

Davis, N., Raina, G., and Jagannathan, K. (2018). Taxi Demand Forecasting: A HEDGE-Based Tessellation Strategy for Improved Accuracy. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12.

Davis, N., Raina, G., and Jagannathan, K. (2019). Grids versus Graphs: Partitioning Space for Improved Taxi Demand-Supply Forecasts. *IEEE Transactions on Intelligent Transportation Systems PREPRINT*, pages 1–10.

De Gooijer, J. G. and Hyndman, R. J. (2006). 25 Years of Time Series Forecasting. *International Journal of Forecasting*, 22(3):443–473.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, 74(366):427.

Diebold, F. X. and Mariano, R. S. (1995). Comparing Predictive Accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263.

Dunning, T. and Friedman, E. (2017). *Machine Learning Logistics*. O'Reilly Media, Inc.

Flores, B. E. (1986). A pragmatic view of accuracy measurement in forecasting. *Omega*, 14(2):93–98.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004a). Learning with Drift Detection. In *Advances in Artificial Intelligence – SBIA 2004*, pages 286–295. Springer Berlin Heidelberg.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004b). Learning with Drift Detection. In Bazzan, A. L. C. and Labidi, S., editors, *Advances in Artificial*

Intelligence – SBIA 2004, pages 286–295, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37.

Gonçalves, P. M., De Carvalho Santos, S. G., Barros, R. S., and Vieira, D. C. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156.

Guajardo, J. A., Weber, R., and Miranda, J. (2010). A model updating strategy for predicting time series with seasonal patterns. *Applied Soft Computing Journal*, 10(1):276–283.

Holt, C. C. (1957). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10.

Hyndman, R. J. and Athanasopoulos, G. (2013). Forecasting: Principles and practice. `https://otexts.com/fpp2/`. Last accessed on: 2019-5-16.

Jiang, W. and Zhang, L. (2018). Geospatial data to images: A deep-learning framework for traffic forecasting. *Tsinghua Science and Technology*, 24(1):52–64.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Ke, J., Zheng, H., Yang, H., and Chen, X. M. (2017). Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C: Emerging Technologies*, 85(June):591–608.

Kuncheva, L. I. and Žliobait\.e, I. (2009). On the Window Size for Classification in Changing Environments. *Intell. Data Anal.*, 13(6):861–872.

Lewicki, G. and Marino, G. (1989). Approximation of functions of finite variation by superpositions of a Sigmoidal function. *Applied Mathematics Letters*, 17:1147–1152.

L'Heureux, A., Grolinger, K., Elyamany, H. F., and Capretz, M. A. (2017). Machine Learning with Big Data: Challenges and Approaches. *IEEE Access*, 5:7776–7797.

Liao, S., Zhou, L., Di, X., Yuan, B., and Xiong, J. (2018). Large-scale short-term urban taxi demand forecasting using deep learning. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, volume 22, pages 428–433. IEEE.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The Expressive Power of Neural Networks: A View from the Width. In *NIPS'17 Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6232–6240.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808.

Malekian, D. and Hashemi, M. R. (2013). An adaptive profile based fraud detection framework for handling concept drift. *2013 10th International ISC Conference on Information Security and Cryptology, ISCISC 2013*.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56.

Montiel, J., Read, J., Bifet, A., and Abdessalem, T. (2018). Scikit-Multiflow: A Multi-output Streaming Framework. *Journal of Machine Learning Research*, 19(72):1–5.

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. (2013a). On predicting the taxi-passenger demand: A real-time approach. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 54–65.

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. (2013b). Predicting taxi-passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402.

Oneto, L., Ghio, A., Ridella, S., and Anguita, D. (2015). Learning resource-aware classifiers for mobile devices: from regularization to energy efficiency. *Neurocomputing*, 169:225–235.

Page, E. S. (1954). Continuous Inspection Schemes. *Biometrika*, 41:100–115.

Pechenizkiy, M., Bakker, J., Žliobait\.e, I., Ivannikov, A., and Kärkkäinen, T. (2010). Online mass flow prediction in CFB boilers with explicit detection of sudden concept drift. *ACM SIGKDD Explorations Newsletter*, 11(2):109–116.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, 12:2825–2830.

Saadallah, A., Moreira-Matias, L., Sousa, R., Khiari, J., Jenelius, E., and Gama, J. (2018). BRIGHT - Drift-Aware Demand Predictions for Taxi Networks. *IEEE Transactions on Knowledge and Data Engineering*, PP(1):1.

TLC (2019). TLC Trip Record Data. `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`. Last accessed on: 2019-05-02.

Tockar, A. (2014). Riding with the Stars: Passenger Privacy in the NYC Taxicab Dataset. `https://research.neustar.biz/2014/09/15/riding-with-the-stars-passenger-privacy-in-the-nyc-taxicab-dataset/`. Last Accessed on: 2019-04-27.

Tsymbal, A. (2004). The Problem of Concept Drift : Definitions and Related Work (Technical Report TCD-CS-2004-15). *Computer Science Department, Trinity College Dublin*, 106(2).

Tsymbal, A., Pechenizkiy, M., Cunningham, P., and Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68.

Uber (2011). Uber NYC has launched. `https://www.uber.com/blog/new-york-city/uber-nyc-launches-service/`. Last accessed on: 2019-05-05.

Wang, H. and Abraham, Z. (2015). Concept drift detection for streaming data. *Proceedings of the International Joint Conference on Neural Networks*, 2015-Septe.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101.

Williams, B. M. and Hoel, L. A. (2003). Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *Journal of Transportation Engineering*, 129(6):664–672.

Winters, P. R. (1960). Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3):324–342.

Wirth, R. (1995). CRISP-DM : Towards a Standard Process Model for Data Mining. *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, 14(24959):29–39.

Xiao, J., Xiao, Z., Wang, D., Bai, J., Havyarimana, V., and Zeng, F. (2019). Short-term traffic volume prediction by ensemble learning in concept drifting environments. *Knowledge-Based Systems*, 164:213–225.

Xu, J., Rahmatizadeh, R., Boloni, L., and Turgut, D. (2018). Real-Time prediction of taxi demand using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2572–2581.

Zhang, J., Zheng, Y., and Qi, D. (2016). Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254.

Zhao, K., Khryashchev, D., Freire, J., Silva, C., and Vo, H. (2016). Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In *2016 IEEE International Conference on Big Data (Big Data)*, volume 27, pages 833–842. IEEE.

Žliobaitė, I. (2010). Learning under Concept Drift: an Overview. Technical report, Vilnius University.

Žliobaitė, I., Pechenizkiy, M., and Gama, J. (2016). An Overview of Concept Drift Applications. In *Big Data Analysis: New Algorithms for a New Society. SBD*, pages 91–114. Springer International Publishing.