PREDICTING THE DYNAMICS OF SPATIO-TEMPORAL SYSTEMS BASED ON HETEROGENEOUS DATA SOURCES

Blaž Kažič

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Dunja Mladenić, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Evaluation Board:
Prof. Dr. Aleš Švigelj, Chair, Jožef Stefan International Postgraduate School and Jožef Stefan Institute, Ljubljana, Slovenia
Asst. Prof. Primož Škraba, Senior Lecturer, Member, Queen Mary, University of London, London, United Kingdom
Dr. Blaž Fortuna, Member, Jožef Stefan International Postgraduate School and Jožef Stefan Institute, Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Blaž Kažič

PREDICTING THE DYNAMICS OF SPATIO-TEMPORAL SYSTEMS BASED ON HETEROGENEOUS DATA SOURCES

Doctoral Dissertation

NAPOVEDOVANJE DINAMIKE PROSTORSKO-ČASOVNIH SISTEMOV NA PODLAGI HETEROGENIH PODATKOVNIH VIROV

Doktorska disertacija

Supervisor: Prof. Dr. Dunja Mladenić

Ljubljana, Slovenia, November 2020

This thesis is dedicated to my family,
for their love, endless support and encouragement.

Acknowledgments

I wish to express my sincere appreciation to my supervisor, Prof. Dr. Dunja Mladenić, for her ongoing support and encouragement throughout my PhD studies, for her guidance, motivation, and knowledge. I would also like to thank her and Marko Grobelnik for giving me the opportunity to join their research team and providing me with the possibilities to grow during my studies. I also wish to express my gratitude to the members of my thesis committee: Prof. Dr. Aleš Švigelj, Dr. Primož Škraba, and Dr. Blaž Fortuna, for their thoughtful comments and insightful advice, which substantially improved the overall quality of this thesis. I would also like to thank my colleagues, Jan Rupnik, Luka Bradeško, Luka Stopar, and Luis Rei, for their valuable advice, stimulating discussions, selfless assistance and all the fun we had in the past few years. I gratefully acknowledge the funding by the EU projects MobiS (FP7-318452), SUNSEED (FP7-619437) and the Slovenian Research Agency (N1-0058). Last but not least, I thank my family for their unconditional support throughout my studies and life in general.

Abstract

As urbanisation continues to be a trend, in which centralisation of the population into cities is still growing, the importance of intelligent solutions in mobility is in high demand. Likewise, with the integration of renewable energies into all levels of the electrical grid system and the increasing amount of heavy consumers (e.g., electric vehicles), load management of the power grid is becoming a necessity and an ever greater challenge. In the previous decade, we witnessed the emergence of big data, in which we managed to digitise processes and collect massive amounts of data from our environment. We are now in an era in which all this data, combined with increasing computing power and emerging AI algorithms, can be used effectively to find intelligent solutions to the above-mentioned challenges.

In this thesis we propose a novel light-weight method for predicting users' possible next location by using users' mobility logs collected with GPS sensors. Building on previous work, our models are based on statistical Markov state-space models, but we also include additional derived temporal information in the form of "arrival profiles" and "probability of stay" profiles. This method allows us to extend our model even further by using Monte Carlo simulations that additionally enable the simulation of entire mobility patterns (prediction of several future locations with arrival and residence times). We comprehensively evaluate models' predictions on the basis of several real-world datasets and use various evaluation metrics. Overall, the results show that by comparing our results of predicting users' future location with the current state-of-the-art technology, our proposed method produces comparable prediction accuracy rates, but has superior predictive power visible in higher recall rate.

We also propose a generic methodology for pre-processing, fusion, and modeling of heterogeneous time series data streams. The proposed methodology includes data cleaning, feature engineering, and merges data from different data sources into a common feature vector, ready to be used by a predictive algorithm. The proposed approach was applied to a real-world electricity short-term load forecasting scenario, which we extensively evaluated by comparing different feature sets, predictive algorithms, and methodological approaches. The results of our prediction models in practice are comparable to the current state-of-the-art research.

Povzetek

Inteligentne rešitve v mobilnosti so v času urbanizacije, kjer se populacija centralizira v mestih, aktualna tema. Prav tako z vključevanjem vse več obnovljivih virov, kot so domače sončne elektrarne, ter s porastom velikih porabnikov, kot so električni avtomobili, inteligentno upravljanje električnega omrežja postaja nujnost in predstavlja vedno večji izziv. V zadnjem desetletju smo bili priča dobi velikih podatkov, kjer smo uspeli digitalizirati veliko različnih procesov in zbrati veliko količino podatkov. Trenutno pa smo v dobi, ko uporaba teh podatkov skupaj z naprednimi algoritmi omogoča razvoj inteligentnih rešitev za zgoraj omenjene izzive.

V tej doktorski nalogi predlagamo novo metodo za napovedovanje uporabnikove naslednje lokacije na osnovi njegovih preteklih lokacij, zbranih s pomočjo GPS senzorja. Naši modeli temeljijo na statističnih Markovskih modelih, katerim dodamo časovno komponento v obliki "profilov prihoda" ter "profilov bivanja". To nam omogoča nadaljnjo razširitev modela z uporabo Monte Carlo simulacij, ki dodatno omogočajo simulacijo uporabnikove poti (napovedovanje več prihodnjih lokacij s časom prihoda in trajanjem bivanja). Napovedi ovrednotimo z uporabo različnih metrik na podlagi več zbirk podatkov, zbranih iz realnega življenja. Na splošno naši rezultati za napovedovanje prihodnje lokacije kažejo primerljivo stopnjo natančnosti s trenutno objavljenimi raziskavami, pri čemer imajo naši modeli večjo napovedno moč razvidno kot višji priklic.

V doktorski nalogi predlagamo tudi splošno metodologijo za preobdelavo, zlivanje oz. združevanje in modeliranje večjih količin heterogenih časovnih vrst. Predlagana metodologija vključuje čiščenje podatkov, procesiranje značilk in združitev podatkov iz različnih virov podatkov v skupni vektor značilk, pripravljen za uporabo v napovednih algoritmih. Predlagani pristop je bil v praksi uporabljen za potrebe kratkoročne napovedi porabe električne energije. Napovedi podrobno ovrednotimo, pri čemer primerjamo modele naučene z različnimi skupinami značilk, napovednimi algoritmi, ter metodološkimi pristopi. Rezultati na realnih podatkih kažejo, da so napovedi naših modelov primerljive natančnosti s trenutno objavljenimi raziskavami.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AI	...	Artificial Intelligence
AMI	...	Advanced Metering Infrastructure
ANN	...	Artificial Neural Networks
AR	...	Autoregressive Measurements (feature set)
ARSO	...	Slovenian Environmental Agency
DT	...	Date Time (feature set)
EDA	...	Exploratory Data Analysis
GPS	...	Global Positioning System
IoT	...	Internet of Things
KNN	...	k Nearest Neighbours
LR	...	Linear Ridge Regression
LTLF	...	Long-Term Load Forecasting
MA	...	Mean
MAE	...	Mean Absolute Error
MAPE	...	Mean Absolute Percentage Error
MAX	...	Maximum
MC	...	Monte Carlo Integration Method (algorithm)
MDC	...	Mobile Data Challenge (dataset)
MIN	...	Minimum
MQTT	...	Message Queuing Telemetry Transport (protocol)
MTLF	...	Medium Term Load Forecasting
NN	...	Neural Networks
PCD	...	Personal Collected Data (dataset)
SPD	...	Staypoint Detection (algorithm)
STLF	...	Short-Term Load Forecasting
SVM	...	Support Vector Machines
SVM	...	Support Vector Regression
TM	...	Markov Chain Transition Matrix (algorithm)
TMAP	...	Transition Matrix with Arrival Profiles (algorithm)
VAR	...	Variance
VSTLF	...	Very Short Term Load Forecasting
WC	...	Weather Current (feature set)
WF	...	Weather Forecast (feature set)
WLAN	...	Wireless Access Points

Chapter 1

Introduction

With the proliferation of smart sensors, we are increasingly able to access a vast amount of various datasets which can be exploited to improve our analysis or predictions in a certain domain. This trend (i.e., the Internet of things) has also contributed to shifts in applied machine learning area, changing its focus from offline analysis to online analysis. Real-time analytics and stream-based computing are becoming the new norm for data-driven intelligent systems in several domains. In our work, we focus on two important domains that will have a major influence on humanity (especially in congested cities) in the forthcoming decades: smart mobility and smart grid domains.

Based on the current research, we have identified that in both domains, additional data sources can improve the overall results of our analysis, depending on which data sources we include, how we pre-process the data, and which information we include in the model. Additionally, we have determined that it is important to consider the following issues:

- *Spatio-temporal system:* the evolution of a particular location is essentially a spatio-temporal process such as traffic. We have to be able to integrate a time component into our analysis and extract as much information as possible from it in the most efficient manner.

- *Data sparsity and non-stationarity:* highly unevenly sampled and sparse datasets can cause very monotonous and uninformative predictions. Non-stationarity in the dataset can also make it difficult for the model to distinguish between relevant and irrelevant patterns.

- *Data integrity:* a vast amount of messy and heterogeneous time series data that has to be properly addressed for best results, doing so includes time alignment, merging several data sources with different sampling rates.

- *Adaptive models:* models have to be able to adapt to new data and significant changes in data. This phenomenon is common in real-world applications, also known as *concept drift*.

- *Security, privacy and data integrity:* data, such as GPS logs from smartphones, satellite tolling, and electricity consumption logs represent a new opportunity in smart cities, but we have to consider that these data are also extremely personal; therefore, users' privacy must be taken seriously.

The increasing rates at which data is being collected and the continuous growth of computational power has caused the encouraging environment for the development of new data-driven streaming analytical algorithms that drive these systems. However, these

systems rely on pre-processing pipelines developed specifically for each new application, often from scratch. Despite the fact that data pre-processing has been shown to consume the majority of the effort when developing such systems, there is still no generic pre-processing and fusion system available. Designing an efficient and reliable framework for processing high-velocity streaming heterogeneous data remains an ongoing challenge.

In this thesis, we present a generic framework capable of processing real-world streaming data from sensors. We explore and model the dynamics of spatio-temporal systems, from real-world smart cities use cases, as two essentially different approaches: as a state transition problem and as a multivariate autoregressive problem. We perform an extensive analysis on merging various sources by using different modelling approaches, and analyse how this affects the overall system performance. Finally, we assess if the added complexity of using a streaming approach is justified in terms of model performance on the real-world scenario use cases.

1.1 Aims and Hypothesis

The main goals of the dissertation include proposing an online generic stream processing framework that will be able to process real-world scenario datasets of different properties (static, dynamic, and forecasted data). Furthermore, we propose a new statistical method in GEO spatial analytics for predicting users' next location, based on GPS data and additional temporal features, which goes beyond predicting only one future location.

The main objectives of the dissertation are the following:

- **O1:** Extensive experimental analysis of using heterogeneous data sources and how this affects the models' overall performance;
- **O2:** Compare online algorithms vs batch vs offline processing pipelines and compare its effect on models' overall performance;
- **O3:** Propose new methods for next location prediction, including additional temporal features and using Monte Carlo simulations, which enable simulating human mobility patterns;
- **O4:** Evaluate the framework and predictions resulting from its application on real-world scenario use cases (traffic prediction, load forecasting).

The thesis hypotheses are two-fold:

- **H1:** Including multiple relevant data sources improves the overall model performance (e.g., accuracy, recall) in predictive time series analytics;
- **H2:** In real-world scenario applications, online streaming models enables better application performance and produce comparable prediction results as offline models.

1.2 Contributions to Science

The proposed thesis advances the field of advanced analytics for smart cities by proposing new algorithms for processing large amounts of heterogeneous data sources, on real-world data from two different fields: *smart mobility* and *smart grids*. We analyse how including different data sources affects the prediction performance of the model. We hypothesise that more data sources should increase the accuracy of the prediction, but we also expect

that some additional data might also worsen the predictions (due to poor-quality data or unrelated information that might confuse the algorithm when the amount of data is insufficient). We anticipate that pre-processing, such as merging methods, resampling rates, and feature engineering, will have an even greater effect on model performance.

Since very few limited frameworks for stream pre-processing were found in the existing literature, we propose a generic streaming processing framework able to process real-time streams of data from multiple sources. In practice, developing, using, and even maintaining an online streaming system is a much more complex task than using an offline trained system. Therefore, we also compare the performance of offline, batch, and online models. Many researchers claim that the online models perform comparably well to offline models, but their experiments are usually done in a controlled environment on synthetic data, or already pre-processed feature vectors. We compare the results by using real-world scenario data and evaluate the results, considering the trade-off between system complexity and performance in practice on real-world use cases.

In the field of smart analytics for smart cities, we present a new algorithm used for GEO spatial analysis (i.e. next location prediction), using the GPS data logs from an individual, enhanced with temporal features derived from the mobility logs. The idea to include temporal information to improve accuracy of predicting the next location is not new, but we incorporate it in a fundamentally different way: in the form of arrival profiles, which is well-suited for highly unevenly sampled datasets. Furthermore, we extend the model with Monte Carlo integration simulations, which additionally enables predicting also the most probable time of location transition and duration of stay. To the best of our knowledge, this is also the first model that enables predicting human mobility patterns (personal life routines), meaning that the model can predict more than one location into the future, including residence time. For example, this allows us to predict multiple locations in the future (including times of arrivals and departures), as well as to answer complex questions regarding individuals' mobility patterns.

The main scientific contributions of the dissertation are:

- **C1:** A new light-weight method for predicting next location from GPS data logs, enhanced with temporal features (TMAP, see Section 2.3.2), which improves models' predictability (i.e. recall, or also called sensitivity);
- **C2:** Extension of the proposed new method based on Monte Carlo simulations (MC, see Section 2.3.3), which goes beyond next-location prediction and is able to simulate entire human mobility patterns;
- **C3:** Extensive experimental analysis on real-world scenario use cases (i.e. smart mobility in Section 2.4, smart grid in Section 3.3) on how using different heterogeneous data sources and various predictive algorithms affects the overall models' performance.

1.3 Overview of the Thesis

This thesis is structured as follows. In Chapter 1, we first present the motivation behind our research and introduce the main aims and hypotheses to the reader, followed by stating the main contributions of our thesis to the field and science in general.

We continue with the two main chapters of our thesis, where we present our research work on two conceptually different approaches: State Transition Modeling (Chapter 2) and Time Series Regression Modeling (Chapter 3). In both chapters, we first present the related

work and position our work in the state-of-the-art of the respective field, by comparing our methodology, results, and conclusions (Section 2.1 and Section 3.1). We continue by introducing the methodology (Section 2.2 and Section 3.2) and proceed with extensive analysis and evaluations of our proposed methods (Section 2.4 and Section 3.3.6). We conclude each chapter with a presentation of practical application that was developed by using the proposed approach (Section 2.5 and Section 3.4) to demonstrate the performance and practical benefits of our proposed methods in real-world use case scenarios.

In the concluding section (Chapter 4), we discuss our hypothesis stated in the introduction chapter and consider the possible improvements and directions for future work.

Chapter 2

State Transition Modeling

State transition modeling is a statistical approach that includes both Markov model simulation and Monte Carlo Integration micro simulations [1]. The model describes the dynamics and statistically possible transitions between the states, commonly used in predictive modeling and probabilistic forecasting in many real-world applications from various fields (clinical decision analysis, credit score analysis, risk factor assessments, forecasting price trends, etc.).

In this chapter we use state transition modeling to present the novel algorithm for predicting users' next location based on their prior movements and based on the timestamp features (see Figure 2.1). In our case the states are considered as individuals' locations or *staypoints* – location where the user stayed for a longer period of time. We refer to this as *next location(s) prediction*. Note that we do not restrict ourselves to only the next location, but consider the problem of predicting likely locations over any future time window.

Accurate location-related information enables applications the ability to provide relevant information, such as smart traffic notifications, suggesting nearby social events and enabling target advertising. By knowing the user's current location, and proactively predicting the user's movement patterns, traffic notifications can be tailored to their location at specific time(s) and mobile assistant applications can provide additional suggestions based on likely future locations. For example, the user can be informed a day in advance about road closures on the route to work he or she usually takes each morning. Another example is a "location-aware" target advertisement. Smart ads could provide even more personalized ads, relevant to the users' mobility patterns, which could increase the success rate of ads. For example, offering a client a new restaurant ad if the next location is predicted to be a restaurant, or offering groceries ads if the next location is predicted to be a grocery store.

At first glance, this resembles the classical problem of learning a distribution over time and frequent locations. There are several important obstacles which make this direct approach infeasible. First, raw GPS coordinates are far too noisy to be used to directly learn movement patterns. To overcome this problem, we pre-process the data into *staypoints* (used as states). This provides a discretization of space into a fixed set of locations, vastly simplifying the computation required for prediction. As the sampling rate of the data is relatively high (1 Hz on average for most smartphones), computationally efficient methods are critical to ensure the problem remains feasible.

Efficiently computing predictions is non-trivial due to the large number of dependencies in the system, i.e. the user's movements are not independent of the user's current location, day of the week, time of day, etc. Handling these dependencies, while maintaining a reasonable model size, is a significant challenge that we address in this research. Highly uneven sampling is another important challenge. Despite a large amount of the available

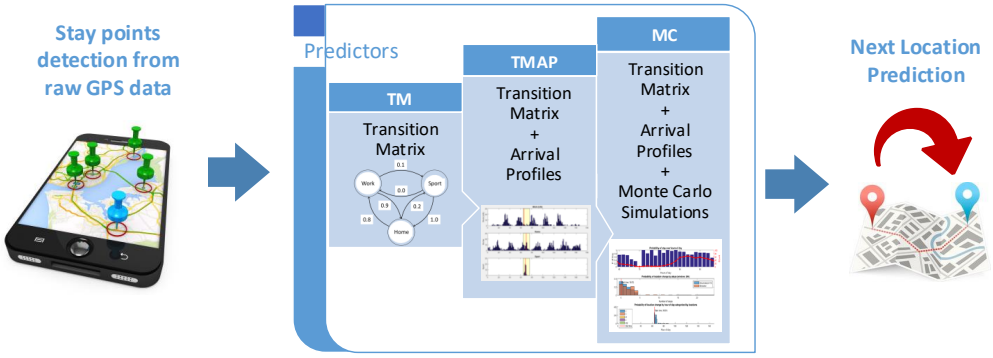


Figure 2.1: For predicting the users' next location, we use Markov model transition matrix as a basis, and then gradually improve the idea by adding additional temporal information and Monte Carlo simulations to improve the performance. The models are used to predict the next location, time of arrival and residence time for multiple jumps into the future.

data, certain locations are heavily sampled (e.g. users' home and/or work), while others are sparsely sampled (e.g. restaurants or doctors' appointments). This complicates the prediction problem, as the user's "standard routine" may be predicted every time.

We address these challenges through the main contribution of this thesis: two novel algorithms, based on statistical state space transition models, which are suitable for handling sparse spatio-temporal (GPS) data—highly unevenly sampled records for different locations. Finally, while our methods are aimed at GPS data and toward predicting individual's movements, the overall framework is general and can be adapted to any prediction problem under highly irregular sampling where there are numerous dependencies.

2.1 Related Work

With an increasing amount of real-time location data available (collected from smartphone through GPS, mobile phone cell towers, or wireless access points (WLAN), the interest of studying human mobility and its predictability from spatial trajectories is consistently increasing over the past decade. There are numerous studies published, therefore we group the studies based on the objectives into two types, i.e. *descriptive approaches* and *predictive approaches*, as proposed by Etter et al. [2]. The work in both approaches uses the same data and tackles related problems, but uses different techniques to achieve different objectives. The main objective of *descriptive approaches* is to study the characteristics and statistical properties of human mobility patterns, which is a very useful information when building the predictive models. However, making explicit predictions and building actual predictors is beyond the goal of these studies, which is the main objective of the studies in the *predictive approach* group. Our research falls into the latter group, but we review both approaches for completeness.

2.1.1 Descriptive approaches

One of the important discoveries in human mobility research was reported in 2006 by Brockmann et al. [3]. The authors report that the distribution of traveling distances decays as a power law, indicating that human mobility patterns are similar to scale-free random walks, one example of which is the Lèvy flight model. In practice, this model suggests that a human movement consists of many short trips (i.e. *flights*), followed by occasional longer trips. The analysis was first done by studying traces of bank notes,

but similar outcomes were later reported by using also other data sources, such as GSM locations [4], and more recently GPS data [5], [6].

By studying geospatial data at the individual trajectories level (using mobile phone traces for 100,000 individuals whose position was tracked for a six-month period), Gonzalez et al. [4] report that in contrast with the random trajectories prevailing in Lèvy flight models, human trajectories show a high degree of temporal and spatial regularity. The study reveals that individuals tends to travel mostly among a few highly frequent locations—with high probability of returning to one—indicating that in general, people follow a simple reproducible location patterns.

Due to the inherited regularity of human behavior, authors in [7] report that there is a potential 93% average predictability in user mobility and that a significant share of predictability is encoded in the temporal order of the visitation pattern. Analysis was done by measuring users' entropy from 45,000 mobile users from mobile carriers. Authors also report that heavy tail distribution also applies for predictability which is correlated with the number of unique locations the user visits (individuals who travel less (small entropy) are more predictable than users with many different locations (high entropy)).

Even though the outcomes of these studies are very informative and useful, making explicit prediction of the users' next location, or designing actual predictors, is beyond the goals of these studies.

2.1.2 Predictive approaches

Pioneering work in predicting the next location by using a probabilistic model from raw GPS data was done by Ashbrook and Sartner [8] in 2003. Authors first present their algorithm for clustering raw GPS data into places and later into frequent locations from which they construct various orders of Markov Chain models to predict the users' next most probable location. The paper discusses the usefulness of predictions, but does not contain an evaluation of the prediction accuracy.

In 2006, Song et al. used the publicly available Dartmouth WiFi dataset [9], which contains the WiFi logs of approximately 6000 users over 2 years, for making explicit predictions of people's future whereabouts [10], [11]. Authors exhaustively evaluate four different next location predictors (Markov-based model, compression-based model, PPM [12] and SPM [13]), using different evaluation measures (accuracy, earliness-lateness, and under/over provision). The results of the analysis indicated that users with highly non-stationary mobility patterns are the main cause of poor performance of predictors. This led the authors to introduce aging mechanisms into the models, improving their performance. Additionally, their experiments showed that low-order Markov predictors performed similarly well, or even better, than more complex predictors, which was also reported in other work [14], [15].

The work up to this point focused on predicting only either the location, or time. In 2011, Scelatto et al. [16] presented an approach based on non-linear time series analysis [17], which is able to predict the users' next location, as well as his/her arrival and residence time—i.e. time spent on that location. Authors report an accuracy performance improvement over Markov-based predictors [11], tested on four different publicly available datasets (Cabspotting [18], CenceMeGPS [19], Dartmouth WiFi [9], Ile Sans Fils [20]). Furthermore, their results show that focusing solely on spatial movements may only be useful for short-term predictions. While considering also temporal information proves to be beneficial for both short-term and long-term predictions.

Another approach based on temporal information was presented by Mathew et al. [21] in 2012. The proposed method first clusters human location histories according to their characteristics (the temporal period in which the visits were made) and then trains a

Hidden Markov Model for each cluster, where location characteristics were used as the observable parameter. The proposed model was tested and evaluated on yet another publicly available dataset from the GeoLife project [22] and reports prediction accuracy of about 13.85%.

In 2013, Etter et al. [2] won the Nokia Mobile Data Challenge contest [23] by blending the results of 3 different predictors: a Dynamic Bayesian Network, an Artificial Neural Network, and Gradient Boosted Decision Trees. The idea behind blending results from 3 completely different approaches is to exploit their diversity. The authors report a relative improvement of up to 4% over the individual predictors. The models were evaluated on the MDC dataset [24] (which we also use for evaluating our proposed models). In order to reduce the problems caused by non-stationarity in the dataset, they used the aging algorithm proposed by [10], although the improvement in accuracy due to this algorithm was not quantified. Unfortunately, authors only presented results of first–order models (stateless models), as this was one of the restrictions of the contest.

In the most recent literature, we can detect a trend toward using additional heterogeneous data source in order to improve the models predictions. Since several human mobility studies showed that predictability is often encoded in temporal information [4], [7], it is natural that temporal information has become the most commonly used additional data source, as it often leads to better prediction accuracy and conceptually more meaningful results. In addition to temporal information, social relationship-based data has also been considered. Reported results differ from [25] who reports no systematic improvement in prediction accuracy, while [26] and [27] report a minor improvement, by approximately 2%. Finally, in order to improve to context-sensitive location prediction reliably, Cho et al. [28] used smartphones to identify transportation mode (classified by exploiting accelerometer, magnetic and orientation sensors) as an additional data source.

Furthermore, some of the latest research also tackles the challenge of data sparsity and having no trajectory history for a specific person (cold start), by creating more general models (using data from a group of users) [29]. This enables predicting the future location even for the users for which we do not have any history, or predicting the location which the user has not yet visited [30], [31]. Even though these approaches have their own advantages, they are fundamentally different than the personalized models. In this research, we focus on the latter.

Due to the ever growing amount of locatio–based data and advances in deep neural networks in the recent years, some of the recent work also incorporates neural networks into their pipeline to improve certain tasks. For example, in 2016, Liu et. al [32] proposed Spatial Temporal Recurrent Neural Networks (ST-RNN) which can model local temporal and spatial context in each layer, improving the model with distance-specific transition matrices for different geographical distances, and time-specific transition model for different time intervals. In 2018, Zhang et. al [33] proposed deep spatio-temporal residual networks (ST-ResNet) able to successfully predict the movement of crowds, citywide, using traffic data (historical trajectories) and external factors, such as weather, holiday events and day of the week.

2.1.3 Final remark

Overall, we observe that Markov models still remains the fundamental building block in most approaches (in our proposed methods as well) in the field of next location prediction. Likewise, the idea to include temporal information to improve prediction accuracy is not new, but we incorporate it in a fundamentally different way—in the form of arrival profiles, which is well-suited for highly unevenly sampled datasets (which is a very common case in human mobility datasets). As stated by Lv et. al in [29], the literature is still separately

focusing on two types of predictions: (1) spatio-temporal prediction (predicting where the user will be at a certain time), and (2) next-place prediction (where the user will go next). In our research we extend our proposed model with a Monte Carlo integration–based method, which enables us to answer both previously mentioned questions. Which, to the best of our knowledge, makes our proposed approach the first model that enables simulating multiple locations into the future with residence times (entire human mobility patterns), which is one of the scientific contributions of our thesis.

2.2 Methodology

2.2.1 Staypoint detection

In order to predict users' future locations (places), we need to understand their past location visits and be able to model mobility habits (patterns) from their past. To some extent, raw GPS data contains this information through the distribution and density of points over time. From this raw GPS data points, we need to extract places where the user has stayed for a longer period of time - so-called *staypoints*. However, the data is noisy and does not explicitly specify where (and how long) the user had stayed. For these reasons, we first employ a staypoint detection (SPD) algorithm, which is able to filter out GPS errors and is able to extract locations and residence time.

One of the basic staypoint detection algorithms was presented by [34] in 2005. We extended this algorithm and significantly improved it by using a second pass clustering (depicted in Figure 2.2), which significantly improves the performance of the algorithm, where there is a lot of GPS error in the dataset [35]. The improvements are similar to [36], but we took a different approach (paper on this problem is in preparation). The algorithm clusters raw GPS data with all its anomalies and inaccuracies into dense groups of points where the user stayed for some time. This is done through the use of two threshold parameters Tt (time threshold) and Td (distance threshold). A group of GPS points constitutes a (Tt, Td) staypoint if all pairwise distances are less than Tdt meters and the time difference between the first and the last point exceeds Tt seconds.

As seen in the sample GPS coordinates in Figure 2.2, the original SPD algorithm would

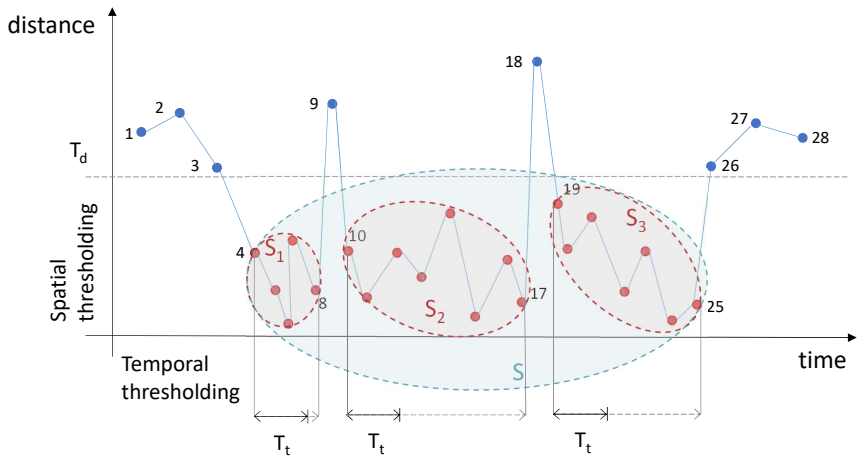


Figure 2.2: Staypoint detection algorithm (SPD) with spatial thresholding Td (distance threshold) and temporal thresholding Tt (time threshold), with a second pass clustering.

detect 3 staypoints (points: $S1\ [4-8], S2\ [10-17], S3\ [19-25]$) and 4 trajectories (points $T1\ [1,2,3], T2\ [9], T3\ [18], T4\ [26,27,28]$). This is not ideal, since two points (9 and 18) are most probably anomalous due to GPS error, so our SPD improves this result by going over the data twice. In the first pass, it performs the standard SPD algorithm [34], then on the second pass it goes over the detected activities (staypoints and trajectories), eliminating $T2$ and $T3$ because their durations are too short ($t < Tt$) and the points afterwards return to the same cluster. This approach merges $S1, S2$ and $S3$ into a single staypoint, thus eliminating the errors. Additional differentiation from the original SPD algorithm is that our approach also keeps trajectories, while original SPD discards all other points that are not part of the staypoints.

The results of our two-pass SPD algorithm are a list of cleaned staypoints (as opposed to raw GPS data) and trajectories of user movement (geo-activities). Besides the location, the descriptions of these activities also contain residence time (start and end time at the location and consequently duration). These geo-activities represent the input data for the prediction algorithm.

2.2.2 Arrival sequence

Once staypoints are extracted from the raw GPS data, we define an arrival sequence.

Definition 2.1. An *arrival sequence* is a sequence of pairs $(L_1, t_1), (L_2, t_2), \ldots, (L_m, t_m)$, where each L_i is a symbol from a set of possible locations $\mathbf{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ and each t_i is a timestamp with the sequence ordered by time. Each pair is interpreted as the fact that the user arrived at location L_i at time t_i.

For example, (*Work*, 2016-05-03 08:00), (*Home*, 2016-05-03 17:00), represents a short arrival sequence, where the user arrived at work at 8:00, stayed there for some time and then started his commute at some time before 17:00 and arrived home at 17:00.

Our hypothesis is that having the temporal information is beneficial to modeling the user's mobility and that daily/weekly commuting patterns can be exploited (e.g. arriving at work at 8:00 on Sunday is less likely than on Monday).

We focus on two inference problems.

1. Predicting the next arrival time and location:

$$P(L_i, t_i | L_{i-1}, t_{i-1}, L_{i-2}, t_{i-2}, \ldots) \tag{P1}$$

2. Predicting the next location when the arrival time is known:

$$P(L_i | t_i, L_{i-1}, t_{i-1}, L_{i-2}, t_{i-2}, \ldots). \tag{P2}$$

The random variables t_i capture the randomness in arrival times and the events of the probability space are arrivals to particular locations at particular times. The model (P1) is able to extend paths, i.e. if we know all the past data, we can generate the next location and arrival time, whereas (P2) is not. For our application, we do not model the prior distribution $P(L_0, T_0)$ (P1 is not fully generative).

2.2.3 Simplifying assumptions

2.2.3.1 Time discretization

We make several assumptions on the conditional distributions in order to simplify parameter estimation. We have already assumed that location data is represented by a discrete

set (the data is obtained by using stay point detection and frequent location algorithms (Section 2.2.1). In order to capture the periodic behavior, we also discretize the temporal information in a straightforward way. We first set the period p (e.g. one week), then select granularity g (e.g. one hour), split the period into $\frac{p}{g}$ bins (e.g. 168 bins) and form the set of bin labels $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_{\frac{p}{g}}\}$. By specifying a start timestamp t_s (e.g. 2016-05-03 00:00), we transform each timestamp t to $t - t_s$ and determine the bin index $i(t - t_s)$ of the appropriate bin (e.g. 2016-05-04 04:45 is mapped to the index 29, since $t - t_s$ equals 29 hours). Formally:

$$g \cdot (i(t - t_s) - 1) + k \cdot p \leq t - t_s \leq g \cdot i(t - t_s) + k \cdot p,$$

for some $k \in \mathbb{N}$. We thus focus on modeling:

$$P(L_i, T_i | L_{i-1}, T_{i-1}, L_{i-2}, T_{i-2}, \ldots), \tag{2.1}$$

where each $T_i \in \boldsymbol{\tau}$ and each $L_i \in \boldsymbol{L}$. Note that since we are modeling arrivals, there is a natural constraint on the location sequence, namely that consecutive locations should be different:

$$P(L_i = \ell_j | L_{i-1} = \ell_j) = 0, \quad \forall i, j.$$

2.2.3.2 Stationarity

We assume that the stochastic process is stationary, i.e. the model does not evolve through time:

$$\begin{aligned} P(L_{i+1} = x_0, T_{i+1} = y_0 | L_i = x_1, T_i = y_1, L_{i-1} = x_2, \ldots) \\ = P(L_i = x_0, T_i = y_0 | L_{i-1} = x_1, T_{i-1} = y_1, \ldots). \end{aligned} \tag{A1}$$

This assumption simplifies the estimation, but may often be violated in practice when longer periods are considered (e.g. work status change, activity starts or ends). This effect is demonstrated in the experimental section (see Figure 2.15 for examples of non-stationarity in real data).

2.2.3.3 Independence

We further impose a first-order Markov structure on the pairs of locations and times:

$$\begin{aligned} P(L_i, T_i | L_{i-1}, T_{i-1}, L_{i-2}, T_{i-2}, \ldots) \\ = P(L_i, T_i | L_{i-1}, T_{i-1}). \end{aligned} \tag{A2}$$

We will also discuss higher-order Markov structures in the experimental section (Section 2.4). An additional Markov-like assumption is that:

$$P(L_i | L_{i-1}, T_i, T_{i-1}) = P(L_i | L_{i-1}, T_i), \tag{A3}$$

which roughly states that the duration of the stay in the last location (L_{i-1}) is not needed when predicting the next location (L_i), given that we know the arrival time (T_i). A third Markov-like assumption is that:

$$P(T_i | L_i, L_{i-1}) = P(T_i | L_i), \tag{A4}$$

which states that the arrival times (T_i) given the next location (L_i) are independent of the previous location (L_{i-1}).

The main consequence of the assumptions is that a complex inference problem (P2) gets highly simplified. As we will show, the parameter estimation will be simplified to estimating three matrices – one for location-location transitions, one for arrival profiles and one for stay profiles.

2.2.3.4 Remark on the assumptions

One of the main simplifications that allows our models to successfully deal with highly unevenly sampled data is the assumption in Equation A4. The idea behind this assumption is that we can build useful arrival profiles for a specific location while ignoring the previous location. In order to test this assumption, we have conducted a statistical test with two different simple models: the null model P_0, based only on arrival profiles $P(T_i|L_i)$, and the alternative model P_1, based on localized arrival profiles $P(T_i|L_i, L_{i-1})$.

Since the alternative model has more parameters and generalizes the simpler null model, we can expect that it will fit more closely to the data. To compare the goodness of the fit, we can compute the log of likelihood ratios between two generative models: the null model:

$$P(T_i, L_i, L_{i-1}) = P(T_i|L_i) * P(L_i, L_{i-1})$$

and the alternative:

$$P(T_i, L_i, L_{i-1}) = P(T_i|L_iL_{i-1}) * P(L_i, L_{i-1}).$$

Both models can be used to generate arrival sequences and assign likelihoods to arrival sequences if the starting location is known. For example, under the null model, knowing L_0, we can sample L_1 using $P(L_1|L_0)$ and from that we can sample T_1 using $P(T_1|L_1)$.

Using the standard framework of statistical testing, we can then check if similar increases in the likelihood can be obtained under the null hypothesis—that is, we fit both models on the dataset and measure the log of the likelihood ratio α. We then use the simpler model to generate similar datasets, each time fitting only the alternative model and each time computing a likelihood ratio, which gives us the distribution of ratios. If α lies in the top 5% of the distribution, we reject the null model with a confidence of 0.05. We experimented with three different sizes of the initial training set: 1398, 500 and 50. The significance level was set to 0.05 and the number of samples to estimate the statistic under the null hypothesis was set to 100.

The statistical test results indicate (see Figure 2.3) that we have to reject our null hypothesis (A4) under all three experimental conditions, which indicates that our alternative model with localized arrival profiles holds more information than our null model. We can also observe that with less data available, the difference between log-likelihoods of the models becomes less significant—and indicates possible over-fitting problems with the richer model. Nevertheless, the space complexity of model P_1 grows quadratically with the number of locations, while the space complexity of model P_0 grows linearly. In order to be able to deal with estimation problems due to high sparsity and also considering the scalability of the approach, we have focused on the validation of the model under the assumption A4, which we will present in Section 2.4.

2.2.4 Models

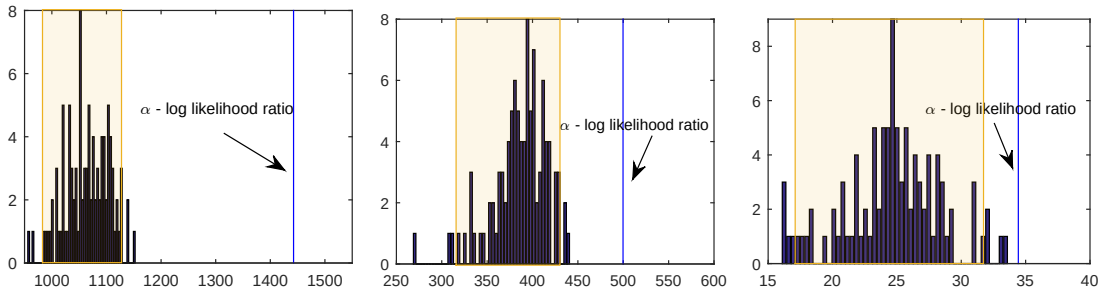
We now present the prediction models related to the assumptions that we have presented.

2.2.4.1 Conditional model (P2)

As a consequence of the assumptions, the problem (P2) is vastly simplified:

$$P(L_i|T_i, L_{i-1}, T_{i-2}, L_{i-2}) \propto \frac{P(L_i|L_{i-1})P(L_i|T_i)}{P(L_i)}. \qquad \text{(P2 + A1,A2,A3,A4)}$$

The quantity $P(L_i|L_{i-1})$ represents the standard Markov transition matrix (with the condition that the location changes) and $P(L_i)$ can be interpreted as the prior arrival location.



(a) Simulations: 100, Records: 1398, Locations: 52

(b) Simulations: 100, Records: 500, Locations: 30

(c) Simulations: 100, Records: 50, Locations: 6

Figure 2.3: Results of statistical likelihood ratio tests for 100 MC simulations show that we must reject the null hypothesis (assumption A4) in all 3 cases. This indicates that the localized arrival profiles contain more information than the general arrival profiles. However, the space complexity of the localized model grows quadratically with the number of locations, while the complexity of model with general arrival profiles grows linearly.

The main temporal information is captured in *arrival profiles* - a $(|\boldsymbol{L}| \times |\boldsymbol{\tau}|)$-table of coefficients representing $P(L_i = \ell_j | T_i = \tau_k)$.

Figure 2.4 shows weekly arrival profiles (represented as arrival frequency histogram over hours of the week) for five common locations from individual's mobility habits, i.e. work, home, groceries and recreational habits. The distinct differences between these locations are clearly seen and illustrate the important temporal information that they contain.

2.2.4.2 Probability of stay

The model (P1) requires an additional ingredient – the *probability of stay model*:

$$P(T_i | T_{i-1}, L_{i-1}). \tag{2.2}$$

This governs when the next arrival will occur (shown in Figure 2.5), or alternatively that the user will stay at location L_{i-1} for $T_i - T_{i-1}$ time units before traveling to a new location. Our approach is based on modeling the event of the staying at a particular location at a given hour of week. We denote by $s_{i,j}$ the probability that the user will be observed at location ℓ_i at time $\tau_j + 1$ given that he was observed at the same location at time τ_j. The probability of stay model is then expressed as:

$$P(T_i = \tau_k | T_{i-1} = \tau_u, L_{i-1} = \ell_j) =$$

$$s_{j,u} * s_{j,u+1} * \cdots * s_{j,k-2} * (1 - s_{j,k-1})$$

for $\tau_k > \tau_u + 1$ and

$$P(T_i = \tau_k | T_{i-1} = \tau_u, L_{i-1} = \ell_j) = (1 - s_{j,k-1})$$

if $\tau_k = \tau_u + 1$. This means that for each time of week and each location, the duration of stay is distributed as a negative binomial distribution with a single parameter. All $s_{i,j}$ parameters form a $|\boldsymbol{L}| \times |\boldsymbol{\tau}|$ matrix of coefficients that have to be estimated from the data.

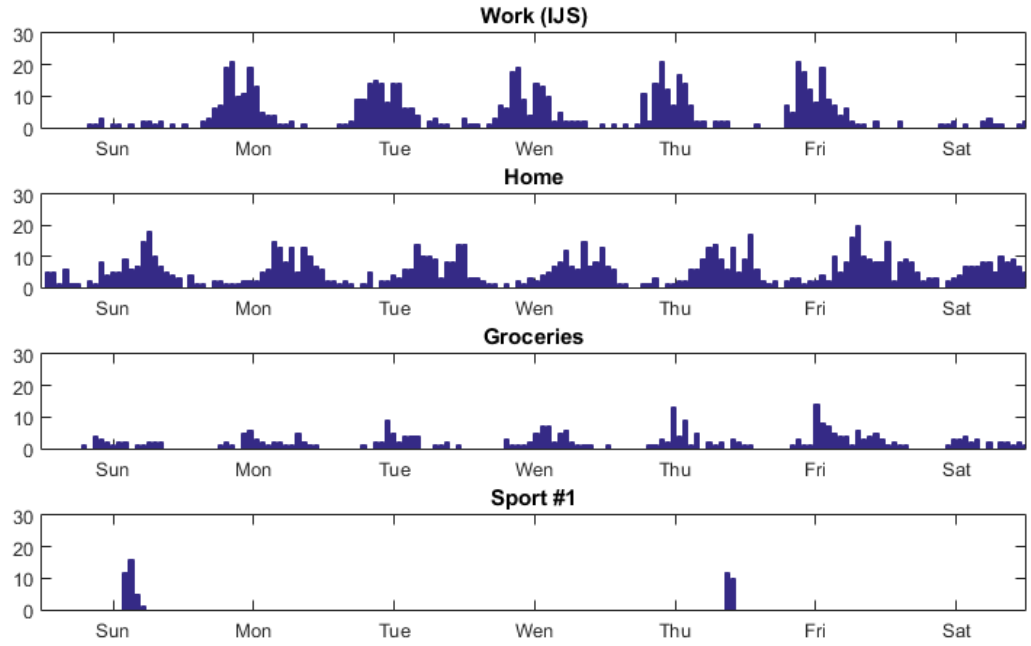


Figure 2.4: Weekly arrival profiles for 4 different locations for an individual. We can observe different weekly and daily mobility habits in the profiles. Profiles encode information on the most probable arrival time to a specific location, based on his history.

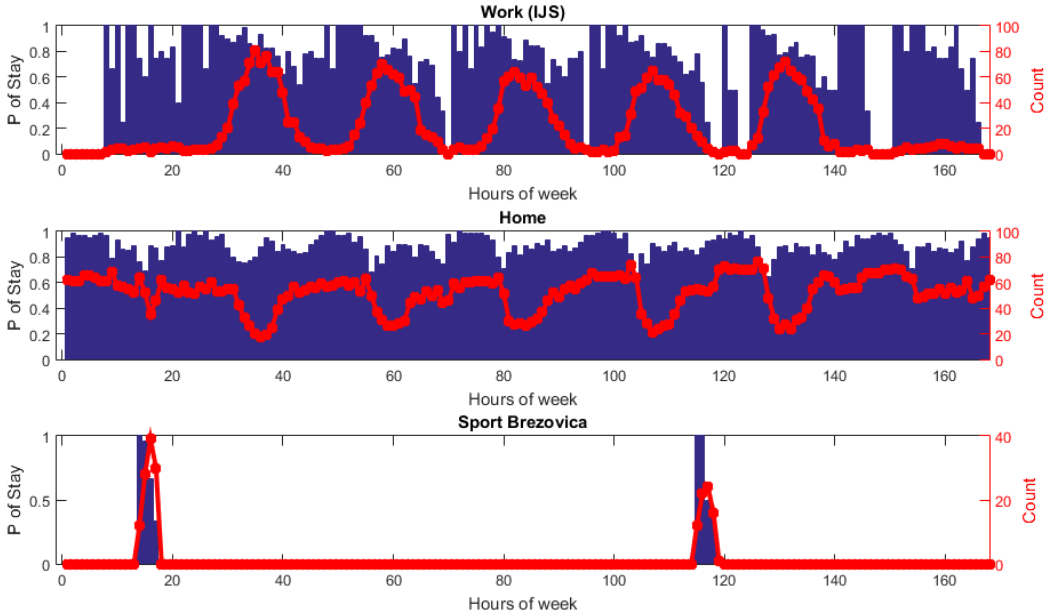


Figure 2.5: Probability of stay profiles for 3 common locations for an individual. These profiles encode the probability of stay at a certain location for a specific hour of the week, based on historical residence times for certain location.

2.2.4.3 Next Location and Arrival Time Model (P1)

Given L_{i-1} and T_{i-1}, we first generate T_i using the probability of stay model, and then generate L_i using the model (P2):

$$P(L_i, T_i | L_{i-1}, T_{i-1}, L_{i-2}, T_{i-2}, \ldots) = P(L_i | T_i, L_{i-1}, T_{i-1}) P(T_i | L_{i-1}, T_{i-1}), \quad \text{(P1 + A1,A2,A3,A4)}$$

where the first factor represents (P2) and the second factor models stay duration.

2.2.4.4 Remark

The main advantage of applying the simplifying assumptions to (P2) is that the number of parameters to estimate is drastically reduced: P1 + A1, A2 requires roughly $|\mathbf{L}|^2|\boldsymbol{\tau}|^2$ parameters, whereas P1 + A1, A2, A3, A4 combined with the stay point model requires: $|\mathbf{L}|^2 + 2|\mathbf{L}||\boldsymbol{\tau}|$ $(P(L_i|L_{i-1}), P(T_i|L_i), P(S_i|L_i))$.

2.3 Predictors

In this section, we present and evaluate three different prediction models (predictors), based on statistical state space transition models, for the purpose of predicting the users' next location, based on his past, current location and time.

We start with the basic, commonly used predictor that is constructed using the first-order Markov Chain method, which enables predicting the next most probable location based on the history of users' movements. We extend the basic method by incorporating temporal information (arrival profiles at the next possible locations), which improves the next location predictions and additionally enables us to predict also the time of the users' movement to the next location. Finally, we propose an approach that models the users' entire mobility based on Monte Carlo simulation. This enables answering more complex time-related questions (such as "What is the most probable location at time HH:MM?"), as well as prediction of multiple locations ahead in the future.

2.3.1 Markov Chain Transition Matrix (TM)

Markov chain models are one of the most commonly used methods in the field of next location prediction. Each entry of the transition matrix represents the probability of the next state (location), estimated based on users' historical transitions (provided as model input, *arrSeq* – arrival sequence). Given the current location (also provided as model input, *startLoc* – start location), in order to find the next most likely location, one only needs to find the location that maximizes the conditional probability using the transition matrix. Probabilities can be presented either as a transition matrix table or in the form of a transition diagram (See Figure 2.6).

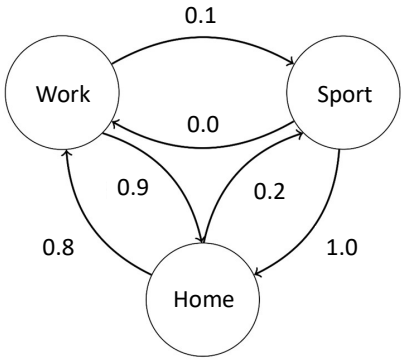
We estimate the transition matrix $P(L_i \mid L_j)$ from the arrival sequence described in Definition 2.1, as the frequency of transitions between location L_i and L_j, normalized by the count of all transitions n from location L_i.

$$P(L_i = l_\alpha \mid L_j = l_\beta) = \frac{\sum_{k=0}^{n} 1 \mid (L_k = l_\alpha, L_{k-1} = l_\beta)}{n} \tag{2.3}$$

For the purposes of prediction, normalization is actually not needed – the next location with the maximal number of visits from the current location will always be predicted. This yields the first baseline algorithm (see Algorithm 2.3).

	Home	Work	Sport
Home	/	0.8	0.2
Work	0.9	/	0.1
Sport	1.0	0.0	/

(a) Transition matrix table



(b) State transition diagram

Figure 2.6: Examples of state transition representation with transition probabilities, for three common users' locations (i.e. *"Work"*, *"Home"* and *"Sport"*). This is a simple model that can predict the next location, based on the current location, but does not take into account the current time. For example, when at location *"Home"*, it will always predict *"Work"* as the next most probable location, even though it is Sunday, when the user usually does not go to work.

Algorithm 2.1: TM (*Transition Matrix*)

Input: $arrSeq$ (arrival sequence), $startLoc$ (current location)
Output: $nextLoc$ (next location)

1. initialization:
   (a) compute *transition matrix* $P(L_{i+1}|L_i)$ (based on $arrSeq$)
   (b) set p (transition vector - *prior*) to be the vector $[P(L_{i+1} = \ell_1|L_i = startLoc), \ldots, P(L_{i+1} = \ell_m|L_i = startLoc)]$ based on current location
2. $nextLoc = argmax(\overline{p'})$ (next most probable location)

This approach is computationally inexpensive, which is its main advantage. Due to this property, the model (transition matrix) for a specific location can be computed "on the fly" (at each query) which is elaborated next. The cost of updating the model per each observed transition is constant as it involves only incrementing a counter. Querying the next location, given the current location, involves simply finding a maximum transition count over a set of previously observed next locations (small set for a single user).

The downside of this approach is that it lacks flexibility, since it always predicts the next location which has the highest transition probability for a particular current location. For example, if the model learned that in most cases, the next most probable location of *"Home"* is location *"Work"*, it will *always* predict *"Work"* as the next location, regardless of the fact that we may have just returned from work. To some extent, we can overcome this drawback ob fist-order model $O(1)$ by using a higher-order Markov model, which additionally takes k-previously visited locations into consideration [10]. Nevertheless, building a higher-order Markov model has the drawback that it also substantially increases the model complexity with the size of the required training dataset, growing exponentially in the order of the model (i.e. k). It also drastically increases computation time, making it too expensive to compute the model "on the fly" at each query. Nevertheless, to assess if the increased model complexity is worth it, in terms of model performance, in the evaluation section (see Section 2.4), we also included the second-order $O(2)$ model (for this and the following two models).

Another shortcoming of the basic Markov chain transition matrix method is that it does not take into consideration any temporal information and predicts the same next location, regardless of the time of day or day of week. For example, for location *"Home"*, it will always predict *"Work"* as the next location, even if it is Sunday. To overcome this, in the next section, we propose a predictor that is also able to incorporate temporal information (e.g. time of arrivals, departures, duration of stay, time of prediction), making the model much more flexible (dynamic) while still preserving simplicity and responsiveness.

2.3.2 Transition Matrix with Arrival Profiles (TMAP)

To overcome some of the drawbacks of the basic transition matrix method (TM) (see Section 2.3.1), we introduce a novel method which incorporates temporal information. In the past, researchers [7] have already recognized that a significant share of predictability is also encoded in the temporal order of mobility patterns (not only in spatial). In the proposed algorithm, temporal information is included in the form of local arrival profile (AP) histograms (see Section 2.2.4.3), computed from the historical dataset, for each specific location (See Figure 2.4). Arrival profiles are computed for each specific location L_i, as an arrival hour histogram for each "hour of week" (7 days $\times$ 24 hours = 168 bins). Note that the arrival profiles themselves could already be used as a simple predictor, giving the most probable location at a given hour of week, without considering the current location. The intuition is that this type of information reveals user's daily and weekly habits, helping with predicting more accurate locations, given the known current time and location. For example, Figure 2.4 reveals that it is a lot more likely that we will go *"Home"* in the afternoon than to *"Work"*.

The main idea for combining the transition matrix (TM) with the arrival profiles (AP) is to use the original transition matrix (as in the basic method in Section 2.3.1) (i.e. *prior*), and update it using the temporal information from arrival profiles to obtain a time-specific transition matrix (i.e. *posterior*). The idea of updating the *prior* prediction with additional information (in our case temporal information) comes from a specific use case of robot localization, where the robot updates its prior knowledge of current location with the additional information of the surroundings from its sensors [37]. The result of

this combination is a more flexible dynamic predictor (TMAP), which adapts predictions to current time.

Additional advantage of this method is that it can also be modified to estimate the time at which the transition will occur. Note that there is relatively little previous work on predicting the time of transition, rather than only the next location (see Section 2.1). The majority of previous work deals only with predicting the next location, with some exceptions such as [16], who used a different approach based on nonlinear time series analysis). The time of transition can be estimated by selecting the maximum values from the arrival profile rows (rather than computing sums) in a sliding time window, defined by the time of query and horizon time (see Algorithm 2.2, step 1.b).

The pseudo-code of TMAP method is presented in Algorithm 2.2. Due to the inclusion of new temporal information, the model has two additional (time-based) input parameters; start time (h)—which is the current time when the prediction is made, and the time window (Δh)—as the size of the time window during which we expect the next transition (usually set between 12 and 24 hours).

The proposed method consists of three main steps:

1. First is the initialization step, where we compute $P(L_i|L_j)$ *transition matrix* (2.3) and *arrival profiles* from the *arrival sequences*. Based on a current location, we set a *transition vector* p which is used as *prior*, and smooth the *arrival profiles* (with Laplacian smoothing).

2. In the second step, we compute the arrival profile weights based on the current hour h and time window Δh. Weights are computed as sums of rows from localized (in time) arrival profiles that are within a prescribed window.

3. In the last step, we update the prior transition vector with the arrival profile weight, and obtain the new transition vector (*posterior*), which now includes temporal information.

Demonstration

The process of updating original transition matrix with additional temporal information (from location-based arrival profiles) is demonstrated on two different example scenarios illustrated in Figures 2.7a and 2.7b, respectively. For demonstration purposes, we take into consideration only a small subset of a transition matrix, with three main locations: *"Work"*, *"Home"* and *"Sport"*. The left side of Figure 2.7 shows the arrival profiles for each location. This reveals the daily commute patterns of our test user. The graphs on the right side present the original (i.e. *prior*) transition matrix (blue), updated (i.e. *posterior*) transition matrix (yellow), along with the weights (green) obtained from the arrival profiles, which were used to update the original transition matrix (i.e. *prior*).

In the first example (Figure 2.7a), we make a prediction at 4 pm, and in the second scenario (Figure 2.7b), at 6 am. For example, if our current location is *"Home"*, we observe that in both examples, our original transition matrix would predict *"Work"* as the most probable next location, no matter the time of prediction. In contrast, for the first example (Figure 2.7a), the posterior transition matrix would predict *"Sport"* as the next location. This is correct, since we can see from the data that our test user usually goes to location *"Sport"* on Tuesday evenings (see Figure 2.4). This is due to the arrival profile graph providing a high weight for location *"Sport"*.

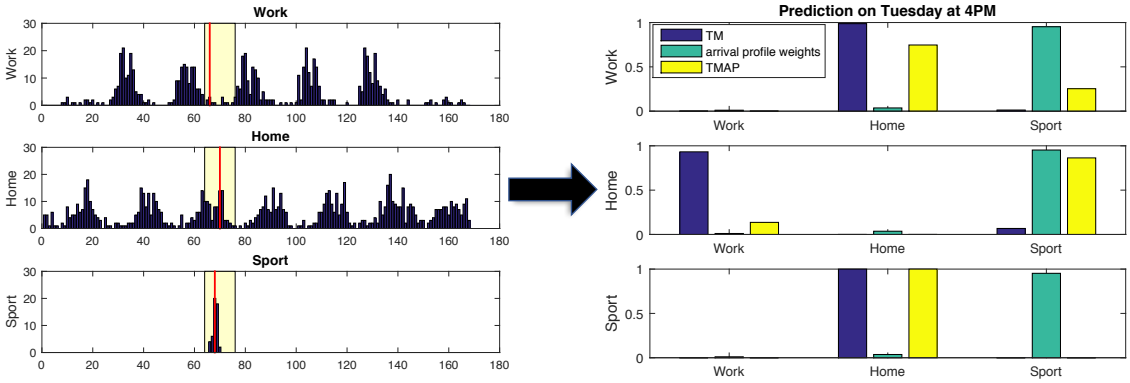
The weights are obtained by summing the bins inside the sliding window (marked in yellow), defined by the hour of prediction and prediction horizon (e.g. 24 hours into the

Algorithm 2.2: TMAP (*Transition Matrix + Arrival Profiles*)

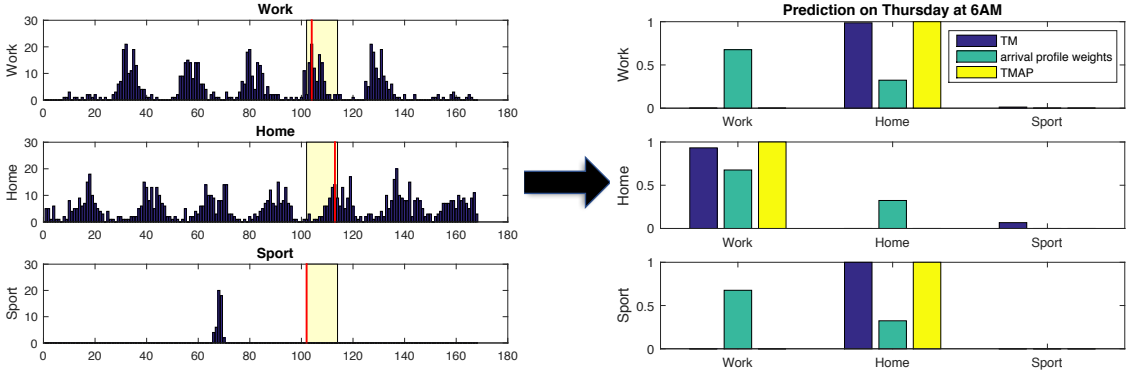
Input: $arrSeq$ (arrival sequence), $startLoc \in \boldsymbol{L}$ (current location), h (start time), Δh (window)

Output: $nextLoc$ (next location)

0. initialization:
   (a) compute *transition matrix* $P(L_{i+1}|L_i)$ (based on $arrSeq$)
   (b) set p (transition vector - *prior*) to be the vector $[P(L_{i+1} = \ell_1|L_i = startLoc), \ldots, P(L_{i+1} = \ell_m|L_i = startLoc)]$ based on current location
   (c) compute weekly arrival profile $A(|\boldsymbol{L}| \times |\boldsymbol{\tau}|)$ (based on input $arrSeq$)
   (d) perform Laplace smoothing on A (with $K = 1$)
1. compute w - arrival profile *weights*
   (a) set A' to be the submatrix with row index $[h, h + \Delta h]$
   (b) set $w = \sum_j A'_{ij}$ (or $w = \max_j(A'_{ij})$)
2. compute new state transition vector p' (*posterior*):
   (a) $p' = p * w$ (update *prior* with arrival profile *weights*)
   (b) $\overline{p'} = \frac{1}{||p'||} * p'$ (normalize *posterior*)
3. $nextLoc = argmax(\overline{p'})$ (next most probable location)



(a) Predictions made on Tuesday afternoon at 4 pm. We observe that both, basic TM and TMAP predictors, estimate that the next location from *"Work"* will be *"Home"*, which is probably correct due to the time of prediction. But given that *"Home"* is our current position, basic TM predicts *"Work"* as the next location, while TMAP uses the available temporal information (arrival profile) and therefore predicts *"Sport"* as the next most probable location, which according to the users' commute habits seen from arrival profile is more sensible for this time of day and day of the week.



(b) Predictions made on Thursday morning at 6 am. The figure shows that the basic TM method produces the same prediction as in the case above (regardless of the different time of prediction), but the TMAP prediction changes with the time of prediction.

Figure 2.7: This example illustrates the main concept of the TMAP method (Section 2.3.2), with weekly *arrival profiles* (for three different locations) on the left, and *transition probabilities* on the right. The original transition probabilities computed from basic first order Markov chain are shown in blue (used as *prior*), the weights derived from arrival profiles are green, and the updated transition probabilities (*posterior*) are yellow.

future). Additionally, we also determine the most probable time of the next transition, which is the hour with the maximal probability weight inside the sliding window. In the second example (Figure 2.7a), prediction was done on a different day at 6 am in the morning. Here, the posterior distribution matrix makes the same prediction as the original transition matrix, predicting *"Work"* as the next most probable location after *"Home"*.

2.3.3 Monte Carlo Integration Method (MC)

We enhance the previous predictor (TMAP) with the Monte Carlo integration method, with the aim of answering even more complex mobility-related questions and predicting beyond only the next location. While the first two methods (Section 2.3.1 and 2.3.2) can answer questions such as *"What is the next most probable location?"*, our goal here is to answer questions of the form, *"What is the most probable location at time hh:mm?"* and *"What is the probability of being at location X at time hh:mm?"*. To answer these types of questions, we have to take into account that the user can change more than one location before arriving to the final location at the desired time. This means that we have to be able to model users' entire mobility patterns, as opposed to modeling only users' location transitions. We achieve this by using the Monte Carlo simulation technique for modeling users' mobility in combination with previously described *Transition Matrix + Arrival Profiles* (TMAP) method (see Section 2.3.2), used for predicting the next location at a specific time in the future.

The main idea of this predictor is to perform multiple mobility simulations, by using the Markov chain model and *probability of stay* (see Section 2.2.4.2) distribution for sampling, in order to obtain an entire distribution of new possible locations in the future. Optimal number of simulations was determined by the convergence properties analysis made for different initial parameters (i.e. various starting locations and starting dates) described in detail in Section A.1 of Appendix A. Given the results of the analysis and also taking into account the computation time (which increases linearly with the number of simulations), 10^3 simulations per prediction were determined as optimal trade-off between performance and computational time. The *probability of stay* for a certain location at a specific time is computed as the ratio between the number of occurrences of staying at a particular location and the number of occurrences of transitioning to another location at a given time (see examples in Figure 2.5).

The pseudo-code of Monte Carlo simulation-based predictor is shown in Algorithm 2.3 and can be divided into the following three main steps:

1. As initialization, we first compute the probability of stay model $P(T_i|T_{i-1}, L_{i-1})$ from the arrival sequence ($arrSeq$). We provide the number of simulations ($iters$) as an input parameter to the model.

2. For each iteration of the simulation we generate a path as an alternating sequence of stays and transitions based on the probability of stay model and the conditional model (P2 + A1,A2,A3,A4). We stop the sequence generation if the time of location change falls out of the time window horizon (Δh parameter).

3. We repeat this simulation for a fixed number of iterations $iters$ and return the distribution of the final location distribution by hour and location (see the lower graph in Figure 2.8a). From this distribution, we can estimate the next most probable locations and times (e.g. by computing maximums from obtained distributions).

Algorithm 2.3: MC (*Monte Carlo integration*)

Input: $arrSeq$ (arrival sequence), $startTime$ (start time), Δh (window parameter), $startLoc$ (current location), $iters$ (number of iterations)
Output: $finalLoc$ (next location), $finalTime$ (final transition hour)

```
// Initialize
```
Estimate the coefficients for the probability of stay model $P(T_i|T_{i-1}, L_{i-1})$ (based on input $arrSeq$)

```
// MC generation of paths consisting of zero or more hops
```
for $counter \leftarrow 1$ **to** $iters$ **do**
    $curTime \leftarrow startTime$
    $curLoc \leftarrow startLoc$
    `// try to make a hop`

    `// Generate the next time of location change`
    $nextTime \sim P(T_i|T_{i-1} = curTime, L_{i-1} = curLoc)$
    if $nextTime > startTime + \Delta h$ **then**

        `// Use the Conditional Model (P2+A1,A2,A3,A4) (sec 2.2.4.1) for next location`
        $nextLoc \sim P(L_{i+1}|T_{i+1} = nextTime, T_i = curTime, L_i = curLoc)$

        `// Override current location and time`
        $curLoc \leftarrow nextLoc$
        $curTime \leftarrow nextTime$
        break
    end
    `// Increment final location and final transition time counters`
    $locations(curLoc) + +$
    $hours(curTime) + +$
    $counter + +$

    `// End current simulation`
end
`// Final location prediction`
$finalLoc = \text{argmax}(locations)$ `// Most frequent final location`
$finalTime = \text{argmax}(hours)$ `// Most frequent final transition hour`

Demonstration

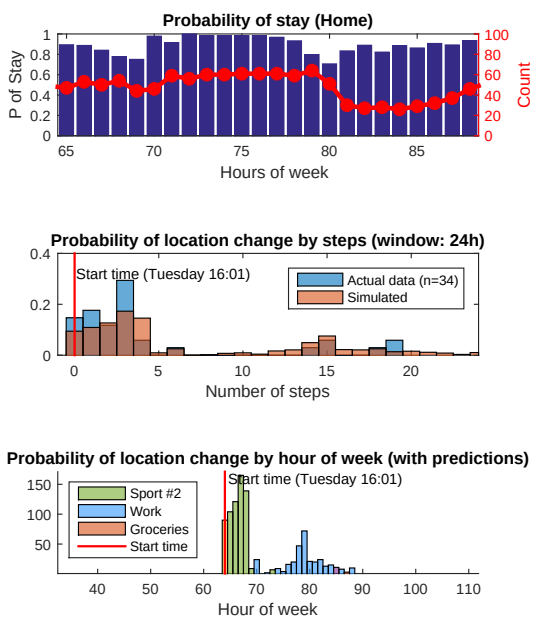
Figure 2.8 demonstrates the MC-based method, with the same examples as in Section 2.3.2. For the prediction from location *"Home"* at 4 pm, recall that the basic transition matrix algorithm naively predicts *"Work"* as the next location even though we usually come back from *"Work"* at this time, while the TMAP predictor more intuitively returned *"Sport"* as the next location since it also took temporal features into consideration, such as day and time of the prediction, in order to produce better predictions. Using the MC method (Figure 2.8a), besides correctly predicting the next location (*"Sport"*), we can additionally estimate the time of location change, as well as the residence time—in this case 7 pm and 3 hours respectively—and even predict more than one location into the future.

The main element that enables us to perform a Monte Carlo simulation is a "Probability of stay" weekly profile, which represents how likely it is that one will stay at a specific location at a certain hour. Probability of stay for this particular problem is presented on the top graph in Figure 2.8, where only a subsection of the entire weekly profile is used (size of this subsection is defined by input parameter δh, with current hour being the starting point). We can see that for location *"Home"*, the probability of stay is higher during the night and lower during the day, which is intuitively what we would expect. By performing multiple transition simulations, we can obtain the entire probability distributions of location changes for the next few hours (the amount of time into the future we consider is the prediction horizon).

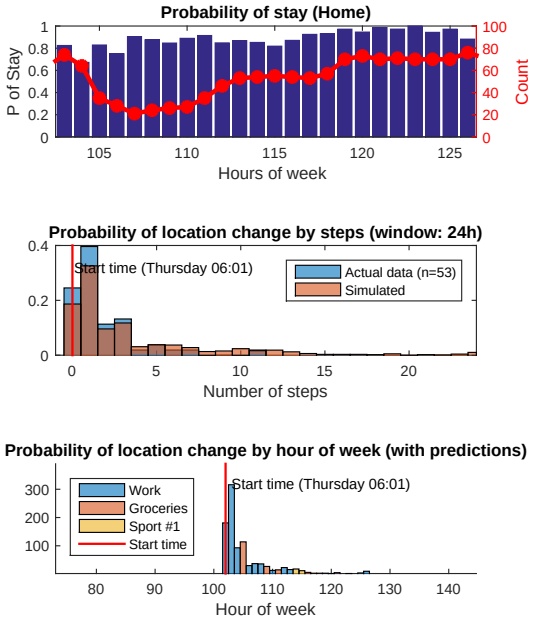
The middle graph in Figure 2.8 shows the location change distribution derived from Monte Carlo simulations, compared to the distribution derived from historical data. We use this graph to test if the simulations look reasonable. We can observe that both—simulated and real data—yield the highest probability of location switch in 3 hours (from the hour the prediction was made – 4 pm), that is at 7 pm.

Furthermore, at each location switch during the simulation process, *Transition Matrix + Arrival Profiles* algorithm is used to predict the next most probable location. Lower graph in Figure 2.8 shows the next most probable locations for the next few hours (based on the current location *"Home"*), along with the probabilities of location change for two known scenarios: Tuesday 4 pm and Thursday 6 am. The prediction results for Tuesday 4 pm (Figure 2.8a) show that for the next three hours (from 4 pm to 7 pm), our model predicts the location *"Sport"* as the most probable location, with the highest probability peak at 7 pm. From 8 pm on, prediction results show *"Work"* as the most probable location, with the highest probability peak at 7 am (and low probability at night). The prediction for Thursday 6 am (Figure 2.8b) shows *"Work"* as the next most probable prediction at 7 am, followed by much less probable location *"Groceries"* at 9 am.

Figure 2.9 additionally demonstrates empirical prediction results also from location *"Work"*, in order to show the models' versatile functionality and practical applicability. The predictions are made at the same time as in the example in Figure 2.8, where predictions were made from the location *"Home"*. Figure 2.9a exhibits the example of prediction from the current location *"Work"* on Tuesday at 4 pm, where the model predicts *"Home"* as the next most probable location in the next five hours, which seems reasonable and consistent with the user' behavior (as can be seen from the profiles in Figure 2.4). While in the second example (Figure 2.9b), the prediction is made on Thursday at 6 am, also from the location *"Work"*, where we can observe that *"Restaurant"* is the most probable next location 5 hours after the query time (11 am), which is reasonable, since this is typically the user's lunch time during working days. The second most probable location is *"Groceries"*, between the hours of 7 am to 10 am, and finally the location *"Home"* after 12 pm has the smallest probability. Recall that this prediction applies to the next location, therefore the fact that *"Home"* has a small probability, implies that it is unlikely that the user will go directly



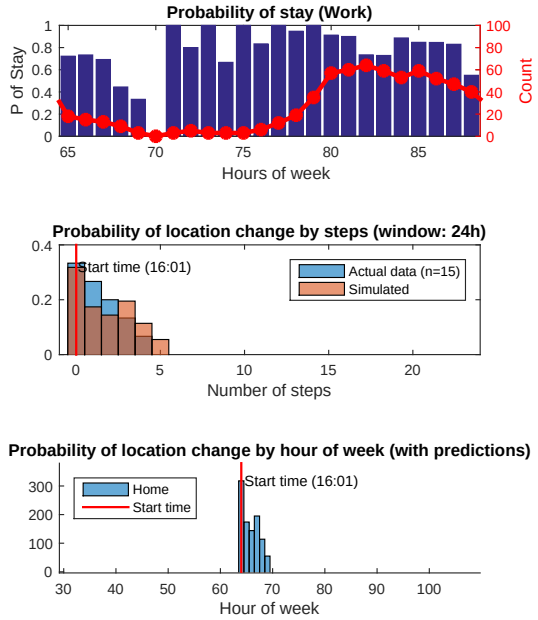
(a) Tuesday at 4 pm, from *"Home"*



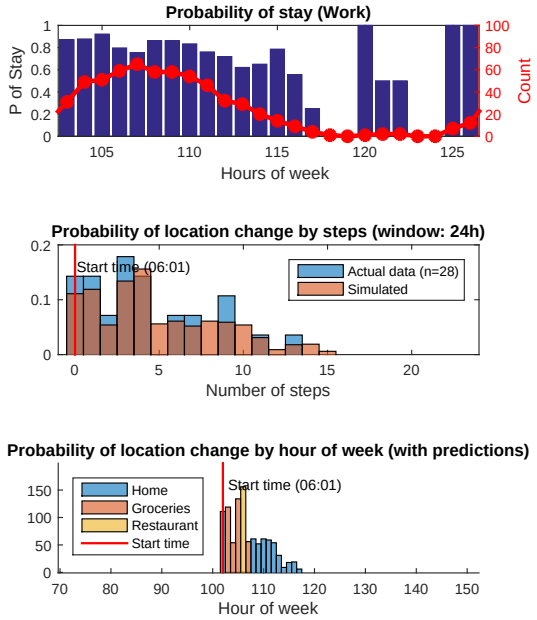
(b) Thursday at 6 am, from *"Home"*

Figure 2.8: An example of MC prediction results from location *"Home"* for the same scenarios as for TMAP (Figure 2.7), i.e. Tuesday 4 pm (Figure 2.8a) and Thursday 6 am (Figure 2.8b). (Top) Subsection of *probability of stay* profile which enables mobility simulation using MC. (Middle) By performing multiple Monte Carlo transition simulations, we obtain the entire location change probability distribution, which is compared to the probability distribution from actual historical data for comparison, to see if the simulations are reasonable. (Bottom) By performing next location prediction at each simulation, we also obtain a distribution of predictions for the next few hours. We observe that for Tuesday 4 pm (Figure 2.8a), the next most probable location is *"Sport"* at 7 pm, followed by *"Work"* at 7 am the next morning. Prediction for Thursday 6 am (Figure 2.8b) reasonably shows "Work" as the next most probable prediction at 7 am, followed by the much less probable location "Groceries" at 9 am.

home without visiting one of the previously mentioned locations first.



(a) Tuesday at 4 pm, from *"Work"*



(b) Thursday at 6 am, from *"Work"*

Figure 2.9: An example of MC prediction results from the location *"Work"*, for the same scenarios as in Figure 2.8, where prediction was made from location *"Home"*. In 2.9a (prediction on Tuesday at 4 pm), we observe that the most likely next location in the next few hours is "Home". In 2.9b, we see the predictions at 6 am on Thursday. The most probable next location is "Restaurant" at 11 am, which makes sense since this is the usual lunch time of the user. After 11 am, "Home" is predicted as the next location with much less probability since it is not very likely that the user will skip lunch and go directly home.

2.4 Evaluation

In this section, we evaluated the performance of the proposed methods on the problem of predicting individuals' next locations. Two real-world datasets are used for evaluation. The first is based on the personal data (PCD) collected by one of the authors, while the second is the publicly available Mobile Data Challenge (MDC) dataset with 170 users [24].

During the entire evaluation process, we use a standard hold out method (90:10), where some part of the dataset (last 10%) is excluded from the training dataset and is used only for testing the performance. To quantify the quality of predictions we used multiple performance measures, including *accuracy*, *recall*, *precision* combined *F1* score. We use micro and macro averaging strategy in order to extend above mentioned binary metrics to multi-class problem, where many mutually exclusive outcomes are possible. See Section A.2.1 of Appendix A for more details. Together, these give a more complete picture of the algorithms' performance.

2.4.1 Evaluation of Personally Collected Dataset (PCD)

2.4.1.1 Data description

Personally collected dataset was gathered during the period from July 2012 to December 2013, by one of the coauthors of [38], using Google's Location History tool. The original dataset contained raw GPS coordinates with UNIX timestamps, therefore the dataset was first preprocessed into a sequence of trajectories and location visits (using the Staypoint Detection method described in Section 2.2.1). Processed data consists of a time-line list of locations (staypoints), defined with their starting timestamp and ending timestamp. After preprocessing, the dataset contains 101 unique locations and 1604 staypoint visits.

Observing the location distribution of obtained data, we can see that the data is extremely heavy tailed (see Figure 2.10), which is not surprising since this is known to be typical for human mobility behaviour [4]. Over 90% of all visits belong to one of the top 10 most frequent locations, which makes this dataset a highly unevenly sampled dataset. A large location bias is one of the main issues when analyzing individual mobility patterns.

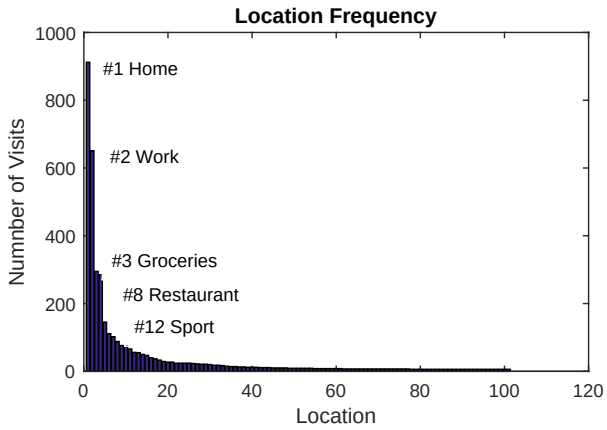


Figure 2.10: Visit frequency analysis histogram shows a heavy tailed class distribution, which is typical for human mobility behavior [4].

Additional analysis of the user's mobility patterns reveals that we are also dealing with an extremely non-stationary dataset. Figure 2.11 depicts the entire collected dataset where each location is colored with a different color. For example, the *Home* location, which is clearly prevailing during night-time hours (from 8 pm to 5 am), is colored with blue, while the *Work* location is more prominent during the day (yellow color). Furthermore, we can observe 2 longer gaps in the data (missing data), as well as several shorter and longer trips. As it was already recognized in the literature ([7]), it is much harder to predict mobility patterns for the individuals who travel often, compared to those with more stationary mobility habits, which is intuitive.

Even though this visualization fails to highlight less frequent location mobility patterns, they can be observed in the arrival profiles (Figure 2.4 in Section 2.2.4.1). Beside the daily mobility patterns (*"Home"* and *"Work"*), we can also observe some periodic weekly habits of the user, such as recreation activities on Tuesdays and Sundays (*"Sport #1"*) and grocery shopping habits which are spread over the entire week, but are most common on Thursdays and Fridays. As we will see later, one of the main advantages of our proposed approach is being able to capture these less frequent habits.

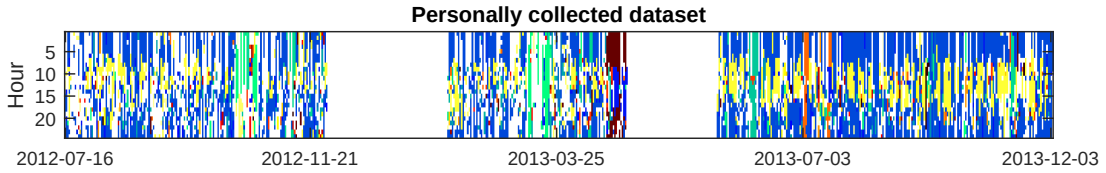


Figure 2.11: Mobility patterns analysis of a personally collected dataset. Figure illustrates main daily patterns (such as location *"Home"* in blue, and "Work" in yellow), but also reveals some non-stationarity in data which pose a challenge for the model. For example, two longer missing data gaps are obvious from the data, as well as a few short trips (orange color) and several longer distance trips over different time zones (green color).

2.4.1.2 Next location prediction

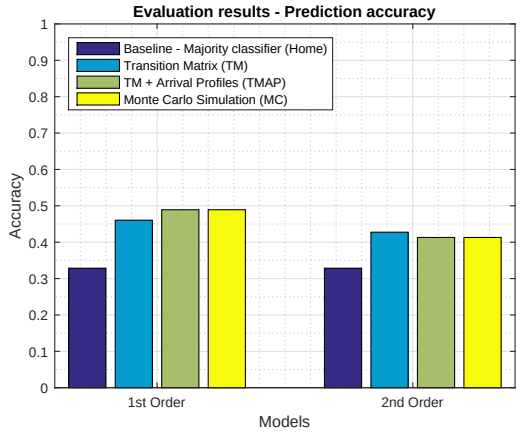
We evaluate the performance of the developed models using different evaluation measures for the most common use case problem from this field – i.e., next place prediction. Since each of the predictors can be developed by using 1st order – $O(1)$, or by using 2nd order – $O(2)$ transition matrices as a basis, we have evaluated both types. In order to put the results more into perspective, by showing the benefit of developed models, we additionally included a commonly used baseline method, which is a simple majority class prediction, usually referred to as "0 order" – $O(0)$ Markov predictor.

2.4.1.3 Overall (micro) accuracy

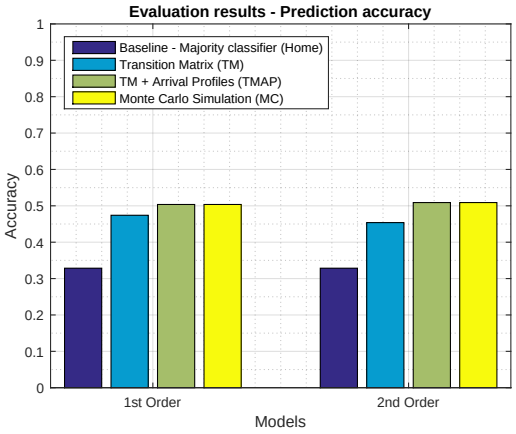
The most commonly used measure to evaluate the performance of next location predictors is the *overall accuracy* measure, also called *micro accuracy* (number of correctly predicted locations across all the predictions). Figure 2.12 shows the accuracy result of the proposed predictors and a baseline predictor. Intuitively, we would assume that the second-order model $O(2)$ will have better accuracy performance, since higher-order models hold more historical information (about previously visited locations). Somewhat surprisingly, the results show no improvement in accuracy when using higher-order models $O(2)$ when compared to the simpler $O(1)$ models (Figure 2.12). However, this is in line with what has also been reported in other work [10], [11], [14], [15].

The two underlying causes for this seem to be non-stationarity and sparsity of data, both of which result in many unseen (or rarely seen) combinations of location transitions in the test set (i.e. transitions which do not appear in the training data). To put it differently, we are much more likely to encounter yet unknown data combinations in the case of $O(2)$ models (since we also take into account the previous location), for which we cannot make predictions, than in the case of $O(1)$ models. Figures 2.12b and 2.12a illustrate the effect of unknown locations on the accuracy scores: Figure 2.12b shows the accuracy where unknown locations were filtered out compared to where no filtering was applied (Figure 2.12a). Naturally, this filtering leads to higher scores, since we have no way of predicting locations which do not appear in the training dataset.

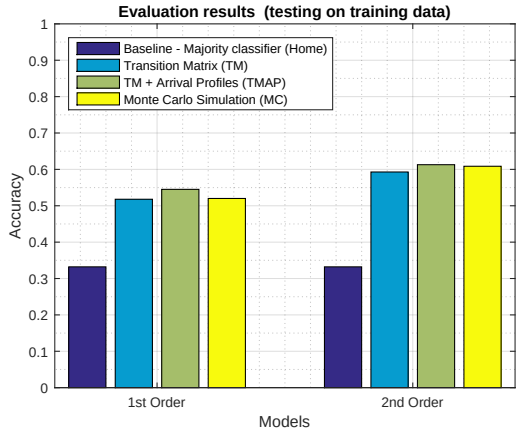
Another confirmation that unseen data combinations reduce the result of $O(2)$ can be seen in Figure 2.12c, where we include the testing dataset into the training dataset (just for demonstration purposes). This way we cannot have any yet unseen data combinations and we can clearly see that $O(2)$ models achieves higher accuracy scores than $O(1)$ models. Similar outcome should also be observed for users who have more stationary mobility patterns and do not travel as much as the user from our personally collected data. Through time, when the model sees more and more data, we should also see the improvement of $O(2)$ which would resemble more the results in Figure 2.12c.



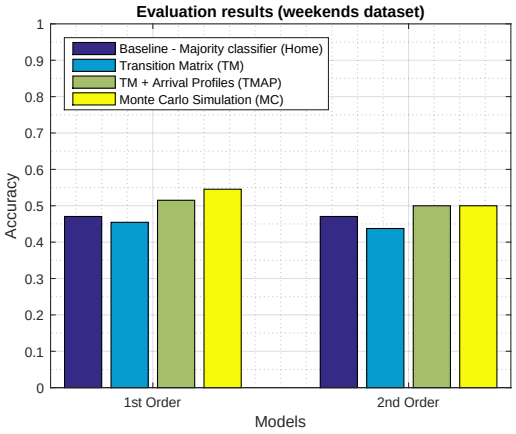
(a) Predictions of unknown locations considered as false predictions



(b) Unknown location predictions excluded from accuracy calculation



(c) Training and testing on the same data



(d) Testing weekends only

Figure 2.12: Micro accuracy comparison between the proposed methods ($O(1)$ and $O(2)$ version) and the baseline, (a) including and (b) excluding unknown locations, respectively. Results show that the main reason for poor performance of $O(2)$ are yet unseen data combinations (of current and previous location) in a testing set. (c) Training and testing on the same data. Accuracy results for $O(2)$ models are significantly better that $O(1)$ models as expected, but may indicate over-fitting. (d) Testing set filtered to weekends only. Normal transition matrix fails to predict meaningfully for weekends—accuracy is even lower than the baseline—while methods that incorporate temporal information (TMAP and MC) perform slightly better indicating that the additional temporal information improves models performance.

2.4.1.4 Accuracy by each class

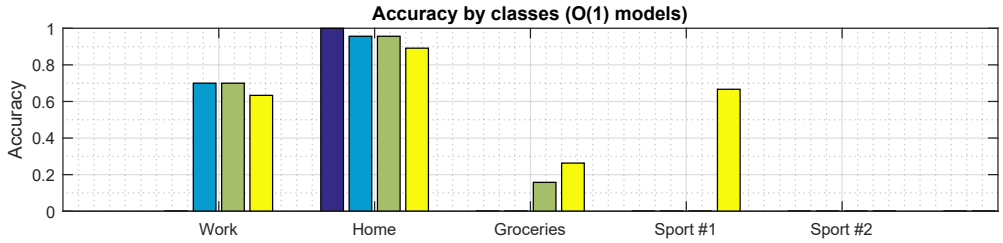
Results in Figure 2.12 show only minor improvements between the proposed models when measuring the overall accuracy over all locations (classes). However, we known that when using micro accuracy, the measure is biased by class frequency. Therefore, using only the overall (micro) accuracy measure bias our models towards doing well only on frequently visited locations, but ignoring less frequent locations. The true benefits of the proposed methods are much more obvious when measuring accuracy results individually by each location (class).

Figure 2.13 shows the accuracy results separately for a few selected frequent locations from the users' mobility patterns. Results highlight the biggest drawback of the baseline methods (majority classifier and TM); they are inherently biased toward predicting only the most common classes—in this case *"Home"* and *"Work"*. In contrast, our proposed methods (TMAP and MC) include predictions also for other, less frequent, but still regularly visited locations such as *"Groceries"* and *"Sport"*. The results show that the second-order TM somewhat reduces this effect (see Figure 2.13b), but it is much more computationally demanding than our proposed methods (TMAP and MC) and requires more data to return meaningful results.

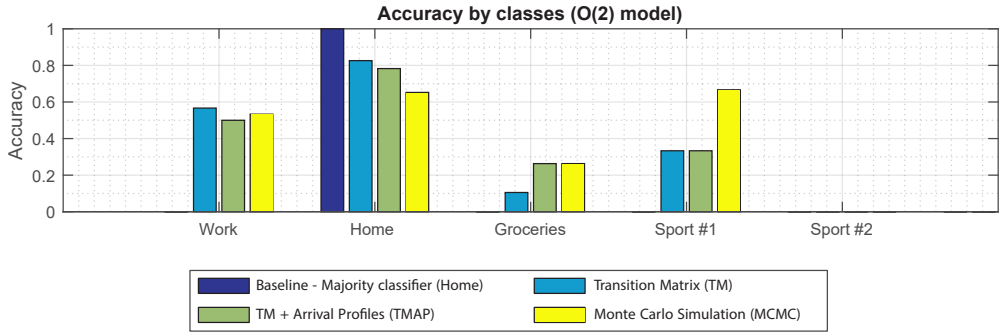
The main reason for improvement is the inclusion of additional temporal information into the model (in the form of *arrival profiles*). As known, a lot of predictability is encoded in the sequence order of place visits (as done in the baseline models), but a significant share of predictability is also encoded in a temporal order of a visitation pattern [7], implying that the models can be improved by inclusion of temporal information (as in our proposed models). This makes the model dynamic, as it makes different predictions for different time of the day, which leads to much broader range of predicted locations, as opposed to basic transition matrix, which considers only spatial information and always predicts the same next location from a certain location, no matter the time (static model).

Figure 2.13 clearly demonstrates that when the user's next location is recreation (*"Sport 1"* or *"Sport 2"*), both methods, which included temporal information (i.e. TMAP and MC), performed better than the other methods (i.e. majority classifier and TM). This is because the weekly arrival histogram (see Figure 2.4) clearly shows that recreation locations are regularly visited twice per week. An improvement in accuracy can also be observed for the location *"Groceries"*, which does not have a narrowly peaked arrival distribution as in the case of the *"Sport"* locations. Rather, it has a spread out distribution during the day, with some small peaks during lunch time and more or less zero occurrences during the night. We observe that such "spread-out" temporal information is still beneficial and improves the accuracy of the model, compared to only relying on previous/current location information.

Additionally, the model also makes conceptually more meaningful predictions. For example, our proposed models do not predict *"Work"* as the next location during the weekends (as it is clear from the arrival histograms that location *"Work"* is not frequently visited during the weekends. Figure 2.12d shows the accuracy scores measured only for weekends. We can see that our methods clearly outperform the baseline methods that use only sequential information about locations. We can see that normal transition matrix during the weekend performs worse than the baseline, while methods that incorporate temporal information (TMAP and MC) perform better than the majority classifier baseline. Although the overall (micro) accuracy is not much higher than the baseline (due to the fact that during the weekend the user mostly goes to location *"Home"* from any location), results indicate that temporal information can improve the performance.



(a) First order models ($O(1)$)



(b) second-order models ($O(2)$)

Figure 2.13: Accuracy for each model class (location). The results show that the majority classifier and TM are biased to predict only the most common classes (in this case *"Home"* and *"Work"*), while the methods TMAP and MC also predicts other, less frequent locations such as *"Groceries"* and *"Sport"*, due to the inclusion of additional temporal information into the model.

2.4.1.5 Additional (macro) evaluation metrics

While the micro accuracy measure (used in Section 2.4.1.3) is informative and easily interpretable, it can also be misleading (biased by class frequency) and is not sufficient for determining if the model is appropriate for a specific problem [39]. In cases where there is a large class imbalance (also called skewed classes)—as is in our case—a model can achieve high accuracy results by predicting only the most frequent class(es) (accuracy paradox [40]). A classic example of such a case is a breast cancer data classifier, where the majority classifier achieves a high accuracy score (since most of the tested patients do not have cancer), but never predicts cancer which is the problem for the unfortunate rare occasion patients who truly have cancer. In our case, such models are not very good at predicting non-frequent next locations, despite achieving high accuracy scores. It may be more desirable to select a model with lower average accuracy if it has a greater predictive power for the problem.

When we are dealing with class imbalanced dataset, we also have to decide (based on the usecase problem), what kind of metric we want to use: *micro-averaged* or *macro-averaged* metrics. While micro averaged metrics are highly biased toward majority classes, macro averaged metrics gives each class equal weight (no matter the frequency). Since we want to design a model with higher predictive power (we want to predict more than just the majority classes–most frequent locations), in this chapter we additionally use macro averaged metrics. Following the evaluation results from previous section (see 2.4.1.4), measuring the accuracy results individually by class (Figure 2.13), only for true positive instances and averaging all the difference class results, this actually means that what we are

measuring is *macro-averaged recall* (also called *sensitivity*), which measures the predictive power of a model.

We thus apply a set of additional macro-average evaluation metrics commonly used in multi-class classification problems (described in detail in Section A.2.1 of Appendix A):

- **accuracy**: rate of correct classifications,
- **precision**: measure of classifiers exactness,
- **recall**: measure of completeness and
- **F1 score**: a harmonic mean of precision and recall.

Results from Figure 2.14 show that in terms of *recall* the proposed methods, which include additional temporal information (i.e. TMAP and MC), clearly outperform the other two (baseline and TM). This implies that the model is able to correctly predict also other classes which are not the most frequent (higher predictive power). On the other hand, with higher *recall* we can observe that the *precision* has declined, which is common for cases with skewed classes. This is because in some cases our proposed models were predicting also other locations, when the actual next location was one of the most frequently visited locations (such as home or work), which was correctly predicted by simple TM, which in most cases predicted only these two anyway. In cases with skewed classes, it is very common to get lower precision if we increased the recall and a trade-off is has to be made. One of the best ways to assess this trade-off is the F1 score. We can see that in our example our proposed models (TMAP and MC) have a slightly higher F1 score than the simpler transition matrix model (TM), for both $O(1)$ and $O(2)$ versions.

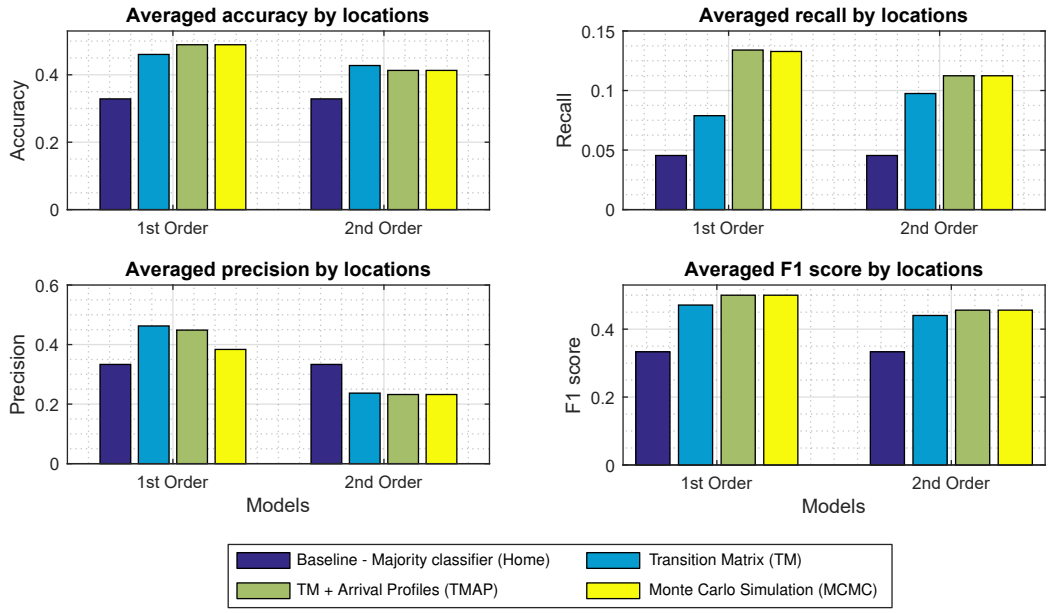


Figure 2.14: Additional macro-averaged evaluation metrics reveal that even though the *micro-averaged accuracy* was not significantly improved by using our proposed method, but *recall* (averaged *accuracy* results by classes, see Figure 2.13) is significantly higher. *Precision* is a bit lower due to higher *recall* measure, which is common for examples with skewed classes, but the *F1* score is again higher (due to increased *recall* measure).

2.4.1.6 Final remark

Given the evaluation results we can conclude that by using our proposed models (TMAP and MC), we achieved fundamentally different prediction results (with higher recall, but lower precision), by predicting also other locations rather than only the most frequently visited ones (higher predictive power). This kind of behaviour can be beneficial in many cases, for example for target advertisement, where it is much more valuable to know (with certain probability) when the user might go to the shopping mall, or a restaurant (even if we are sometimes wrong), rather than only getting predictions for the most frequently visited locations (i.e. home and work) for the price of higher accuracy. In other cases, e.g. for the already mentioned breast cancer example, high precision is much more crucial as we want to predict cancer only with very high confidence.

2.4.2 Evaluation of Nokia MDC Dataset (MDC)

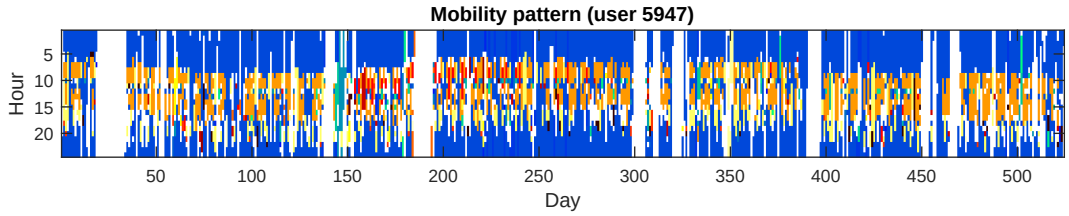
2.4.2.1 Data description

The proposed methods were also tested and evaluated by using a publicly available dataset Nokia Mobile Data Challenge (MDC), which was gathered during Lausanne Data Collection Campaign [24] by 170 students. The main task of the challenge was to predict the next place visited by the user, with some restrictions (due to contest rules), such as user specificity and memoryless predictors [23], and only a subset (50 individuals) of the entire collected data was used in the challenge. The contest was won in 2012 by Etter et al. [2], by combining prediction results from 3 different predictors (Dynamic Bayesian Network, Artificial Neural Network, and Gradient Boosted Decision Trees).

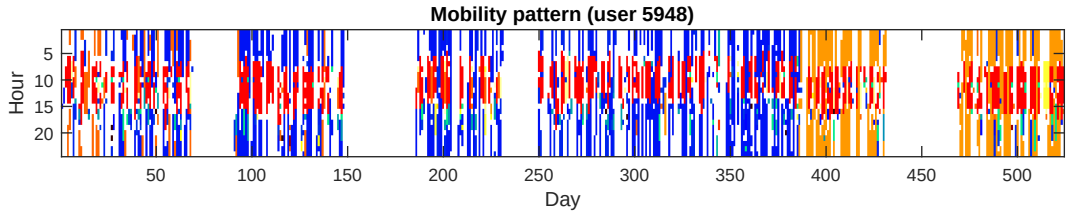
Dataset of sequences of location visits for each participant, where each record contains a unique location id, as well as a starting and ending timestamp, making it comparable to the processed PCD dataset. The number of records per participant varies greatly, from less than 100 records to almost 6000 records, and time-wise from 6 months to 17 months, with various lengths of missing data gaps. Analysing the dataset in detail reveals that for some users we have a very good dataset with very little missing data and clearly consistent daily patterns (see Figure 2.15a). Some users have had major location or lifestyle changes during the collection of the data, which can be clearly seen on the graph (see Figure 2.15b). This is called concept drift, for which the model requires a sufficient amount of data so that it can adapt to changes. For other users, the data shows extremely stochastic and non-stationary mobility habits, combined with large amounts of missing data gaps, (see Figure 2.15c). No clear patterns can be observed due to the highly non-stationary behavior (user moving all the time, or sending a location so rarely that it is not possible to extract any staypoints).

2.4.2.2 Arrival profiles

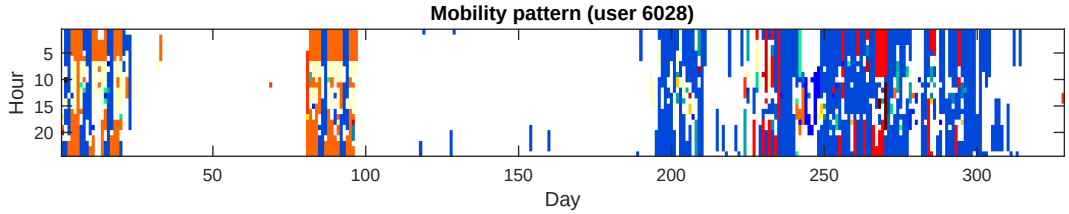
As we incorporate temporal information into our models in the form of arrival profiles, the quality of our proposed models predictions is directly correlated with the quality of arrival profiles, which highly depends on the quality of input data logs. Figure 2.16 shows 4 examples of arrival profiles extracted from data logs of 4 different users, some of them also analysed in the previous section (i.e. user 5947 – Figure 2.15a, user 5948 – Figure 2.15b, user 6028 – Figure 2.15c). Results clearly show that poor quality arrival profiles (Figure 2.16c and Figure 2.16d) are extracted from poor quality data logs (e.g. non-stationarity in Figure 2.15c or concept drift in Figure 2.15b). Such arrival profiles do not add valuable additional temporal information into the model and can therefore even deteriorate the model prediction performance.



(a) High quality data – regular logs with enough granularity.



(b) Concept drift – major location or lifestyle changes require enough data for the model to adapt.



(c) Low quality data – non-stationary, missing data, stochastic with few visible patterns.

Figure 2.15: A mobility pattern analysis of three individuals from the MDC dataset [24]. We observe a large variation in the data quality among these three users.

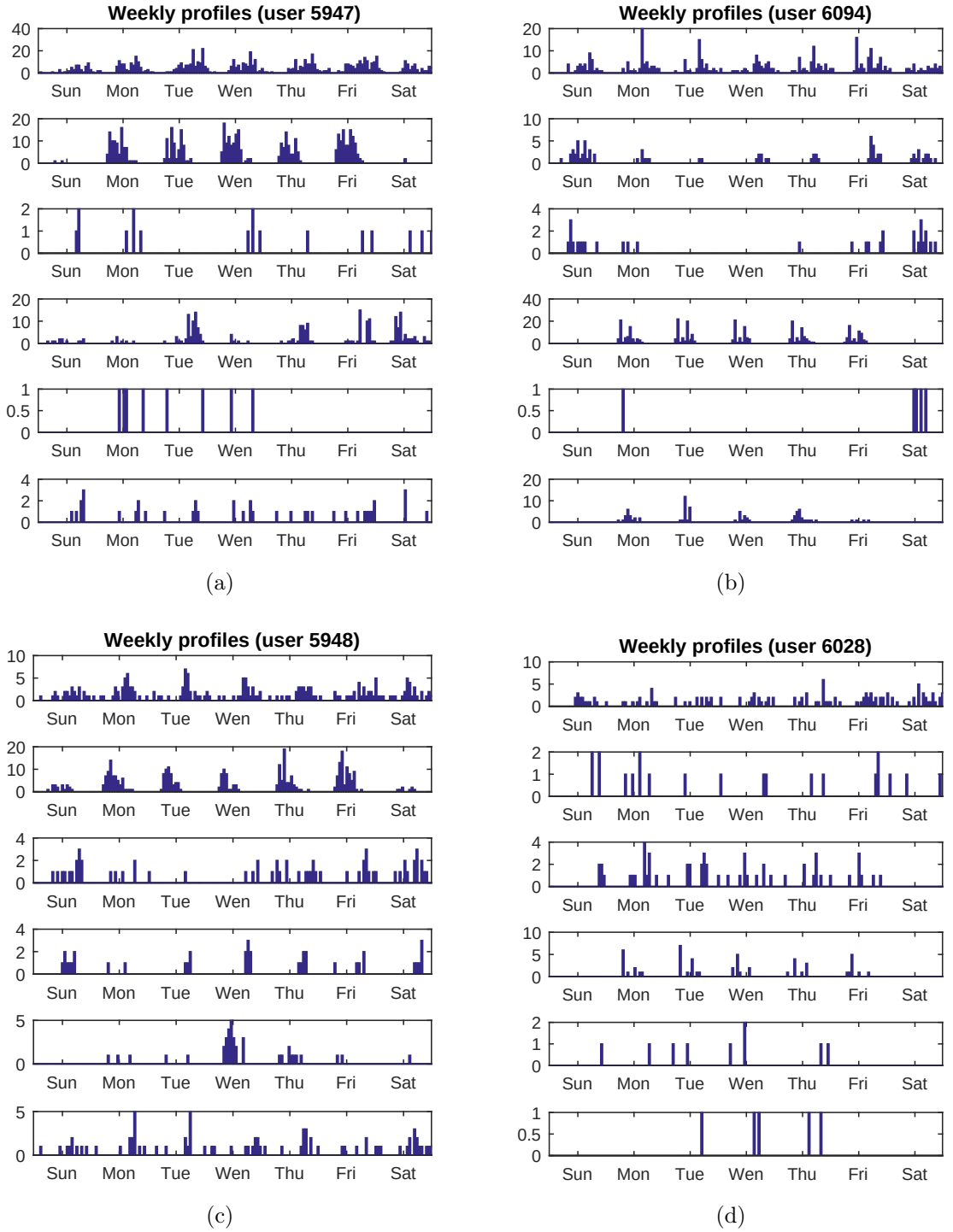


Figure 2.16: (a) Good quality arrival profiles from quality data logs (see Figure 2.15a). Daily as well as weekly patterns can be observed for different locations. (b) Another example of good quality data, where distinct user mobility patterns for different locations can be observed and learned by the model. (c) Example of quality arrival profiles, but with a concept drift (see Figure 2.15b) that can inaccurately update the prior prediction. (d) Example of poor quality data logs (see Figure 2.15c) and arrival profiles. No regular mobility patterns can be extracted from such logs. Such a dataset contains an insufficient amount of information to construct a meaningful arrival profile, which results in low quality predictions.

2.4.2.3 Evaluation Results

Models were evaluated using the same macro-averaged evaluation metrics as in PCD Chapter 2.4.1.5. Since the MDC dataset contains several different users, evaluation was done for each user independently. Averaged results over all users are presented in Figure 2.17. Results show that the best predictor (i.e. TMAP) achieves an average accuracy score of 0.54. This is comparable to the winning model of the Nokia Mobile Data Challenge [2] that achieved an accuracy score of 0.56, by combining results from 3 different models. Similarly, as with the PCD dataset, we can observe slightly better accuracy results when using additional temporal information (TMAP and MC), compared to only using spatial data (TM). Additional improvements of the models using temporal information can also be observed in the recall measure (see Figure 2.17), however, the differences are not as significant as in the case of our PCD dataset. As with the PCD dataset, we can again observe a decrease of precision metrics when using TMAP and MC models, which is again a consequence of larger predictive power (predicting also other locations rather than the most frequent ones). While the F1 score (which is sort of a measure for better assessing the trade-off between precision and recall) is more or less the same for all three predictors.

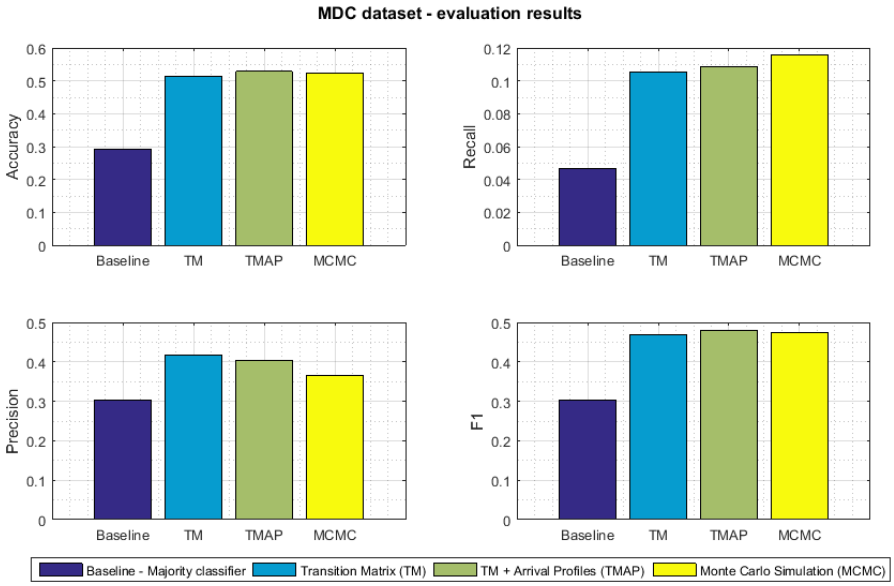


Figure 2.17: Evaluation results for the MDC dataset (macro averaged over all 170 users). Results indicate that proposed methods lead to increased *accuracy* and *recall*. The performance gains are smaller when compared to the PCD dataset. Additional analysis shows that the main reason behind this is poor data quality for some users within the MDC dataset.

Additional analysis shows that the main reason for worse recall improvement than with the PCD dataset is that here we average the results over all 170 users with very diverse quality of input data. In Figure 2.18, we can observe high variances in accuracy scores among individual users, due to the fact that the input data logs for some of the users are clearly of very bad quality (e.g. Figure 2.15b and Figure 2.15c), which consequently leads to high variances in the quality of computed arrival profiles (e.g. Figure 2.16c and Figure 2.16d), as opposed to our PCD dataset, where the data logs are fairly good (see Figure 2.15). For these users, additional temporal information in TMAP and MC models was very poorly extracted, which deteriorates the prediction performance (performing even worse than TM) and lowers the overall average scores presented in Figure 2.17.

Further investigation of users' mobility patterns and arrival profiles revealed the de-

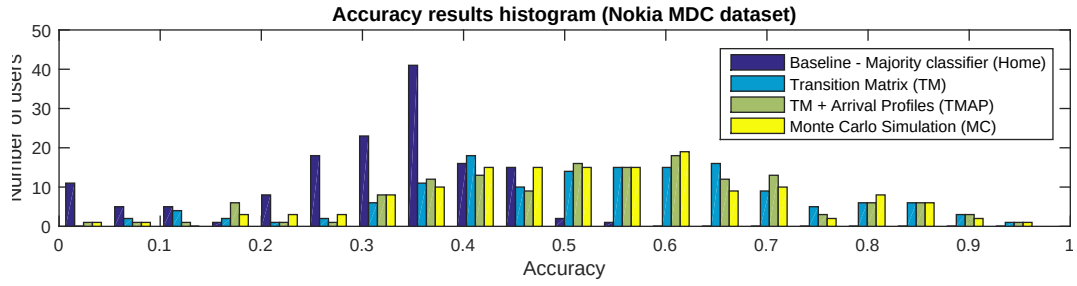


Figure 2.18: Histogram of accuracy scores on the MDC dataset reveals high variance in scores per user. Even though high prediction accuracy was achieved for many users, there are also many users with low accuracy (due to low-quality data logs with consequently lower quality temporal profiles (Fig. 2.16d)).

pendence of our proposed methods on the ability to compute informative location arrival profiles. Figure 2.19 shows the discrepancy between the basic TM method and TMAP method (where TM is updated with arrival profiles). The figure reveals two distinct groups of users: one where using the arrival profiles clearly improves predictions (informative arrival profiles), and one where arrival profiles deteriorate predictions (uninformative arrival profiles). By selecting one of the users from the group where the predictions were improved – user 5947, and by investigating his mobility patterns (Figure 2.15a) and arrival profiles (Figure 2.16a), we can confirm that the key to success was the quality extracted arrival profiles, which successfully updated the prior TM method. On the other hand, since TMAP method is based on updating a basic transition matrix with arrival profiles, poorly extracted profiles results into poor updates, which even worsens the prior prediction. This usually happens when dealing with non-stationary data (Figure 2.15c), or data with a concept drift (Figure 2.15b). An example of poor quality arrival profiles can be observed for user 6028 in Figure 2.16d.

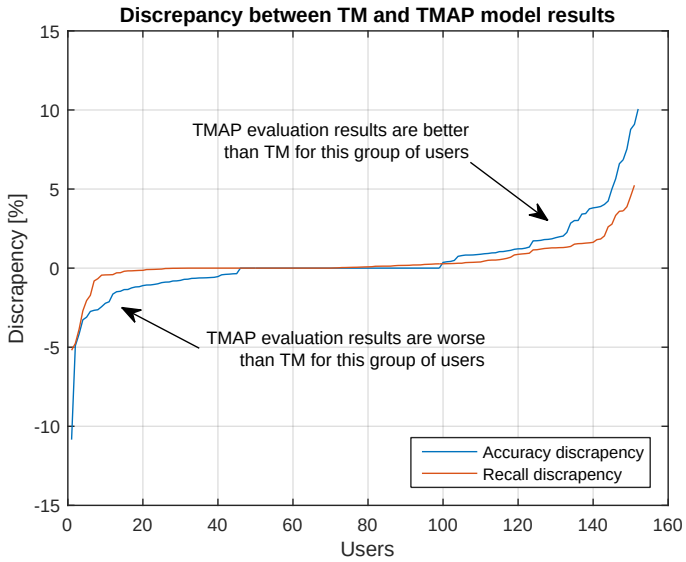


Figure 2.19: Discrepancy graph between the results of the Transition Matrix (TM) model and Transition Matrix with Arrival Profiles (TMAP). We see two distinct groups of users; one with informative arrival profiles where TMAP outperforms TM predictions, and one with poor quality arrival profiles where TM predictions under perform.

2.4.2.4 Final remark

As a final remark, similar observations regarding high variance in the results between users were also reported in the winning paper of the MDC challenge [2]. The authors raised the question if this unpredictability is mainly rooted in the users' personality, or if it is a consequence of the data characteristics. According to our analysis, we conclude that the reason for unpredictability lies in both; see the concept drift example in Figure 2.15b and the non-stationarity example in Figure 2.15c. The difference between these is that the first (concept drift) could be overcome by using more adaptive models (e.g. using an online learning-streaming approach), but it is less clear how to overcome the problem of poor data quality. We conclude that in order to be able to take full advantage of our method, one must possess enough high-quality data so that good arrival profiles can be computed for at least the most frequent locations. This also puts a lot of pressure on the quality and robustness of the SPD algorithm (see Section 2.2.1), which transforms raw GPS data into staypoints, used as input data in prediction models.

2.5 Application

The methods described in this chapter were used to develop a real-world web application for mobility analytics, called NextPin[1]. Application enables real0-time processing raw GPS data (collected by Google locations or by using our mobile app) into staypoints (see Figure 2.20), with automatic transportation mode detection using Random Forest (such as walking, cycling, bus, etc.) [41]. In addition, basic location statistics are computed for the selected time window, such as arrival profiles, location visits by hour of day, average stay at location, etc. (see Figure 2.21). These types of statistics are useful for classification of the location (e.g. home, work, groceries, etc.) and for predictive modeling (already described the usage of arrival profiles in previous sections). External APIs (e.g. Foursquare) are used for additionally labeling the staypoints (title, address, location type, etc.). Staypoints and trips can also be edited (start time, end time, title, modality type, etc.) by the user. Finally, next location predictions (with probabilities) are also visually presented in the application for each specific staypoint (location).

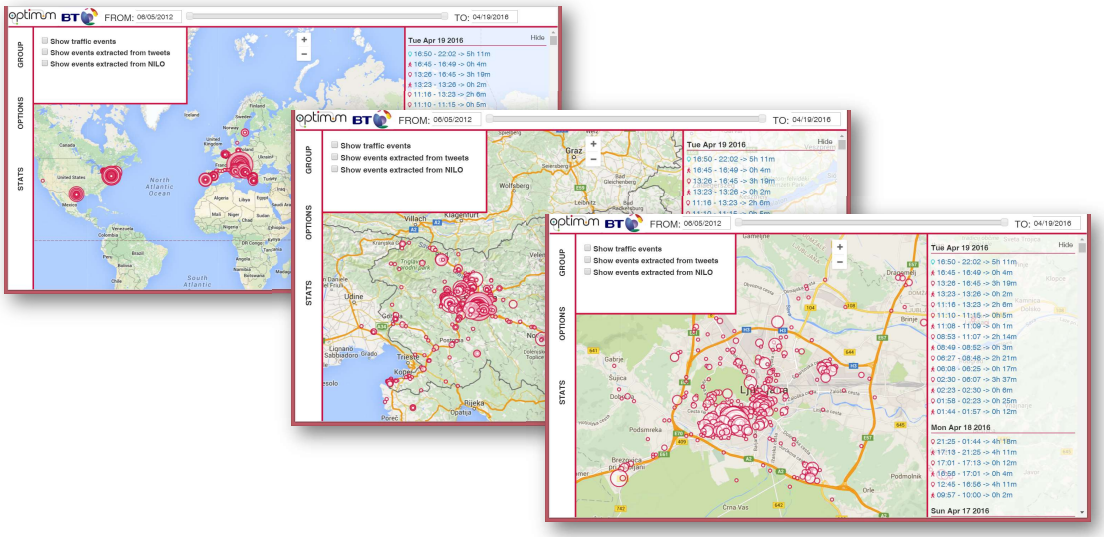
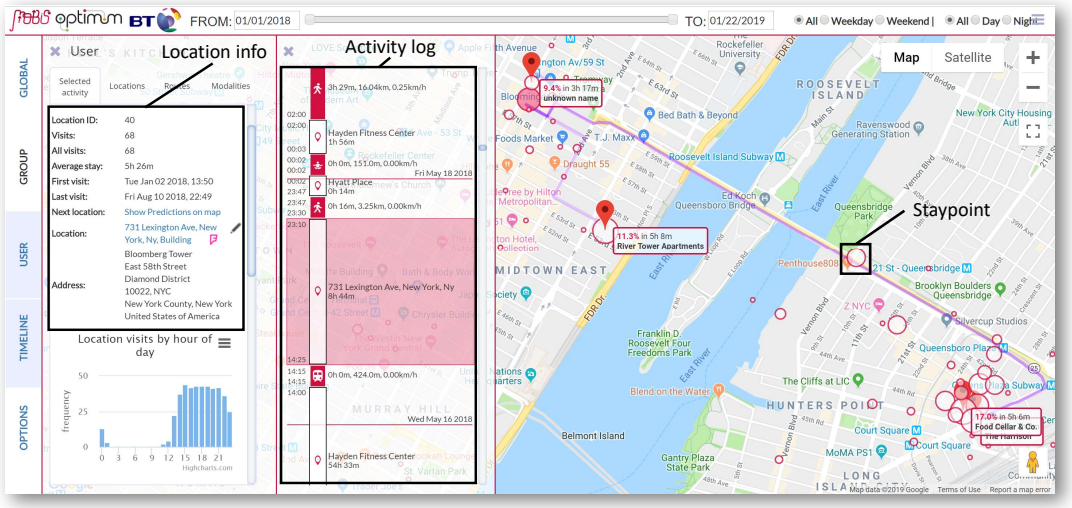
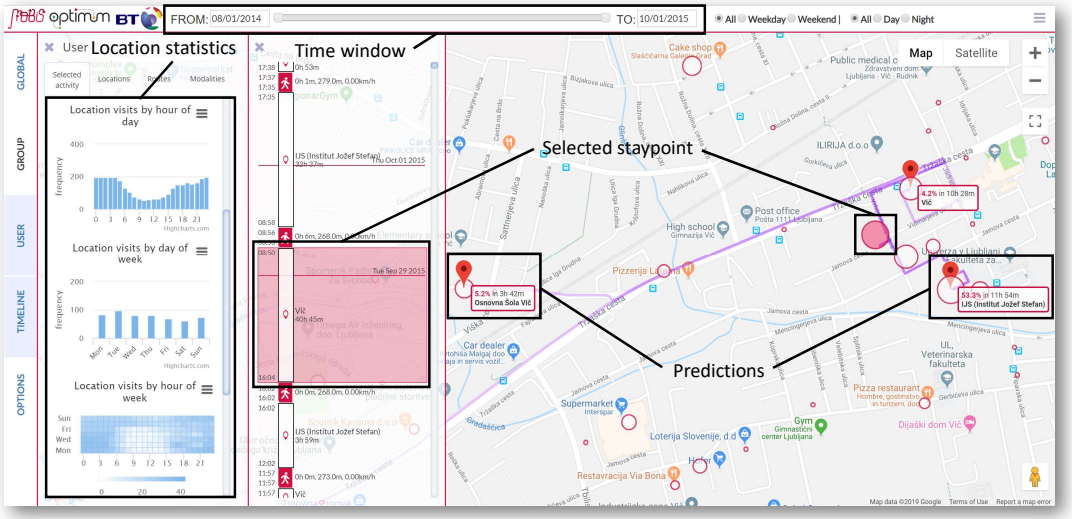


Figure 2.20: Detecting staypoints from raw GPS data in a NextPin web application.

[1] http://traffic.ijs.si/NextPin



(a) Screenshot of mobility analytics from New York (work location selected)



(b) Screenshot of mobility analytics from Ljubljana (home location selected)

Figure 2.21: NextPin analytics view. Statistic properties of a selected staypoint (location) can be explored. Activity log is visualized, along with detected transportation mode. Additionally, most probable next location predictions can be explored for the selected staypoint.

A mobile application called Mobility Patterns (Andorid[2] and iOS[3] version) (see Figure 2.22) was developed in order to acquire data from the user and also enable some basic functionalities of the web application. These are visualizing historical log of daily activities with some basic statistics and showing the top three next location predictions (with probability percentage estimated time of location switch). As a side product, analitical library for Android and iOS development was also released, which consists of the useful mobility algorithms such as staypoint detection and predictive models. This library enables mobile developers to easily integrate and use advanced mobility algorithms and functionalities directly on the mobile phone (without any calls to a remote server).

[2] `https://play.google.com/store/apps/details?id=net.nextpin.example.collection`
[3] `https://itunes.apple.com/us/app/mobility-patterns/id1073388328`

Beside single user analytics, application also enables fleet analytics, i.e. performing grouped staypoints and creating statistical analysis based on entire trips. For example, we had a real-world scenario use case, where the owner of a truck fleet was interested in finding out at which location their fleet was the most frequently, or at which location they spent the most time. They used next location predictions for better logistic planning. Additionally, an anomaly detection system was also developed for the client, which raise alerts for events such as; a particular truck spent more or less time at a certain location as usual, or the truck driver stopped at a new unknown location.

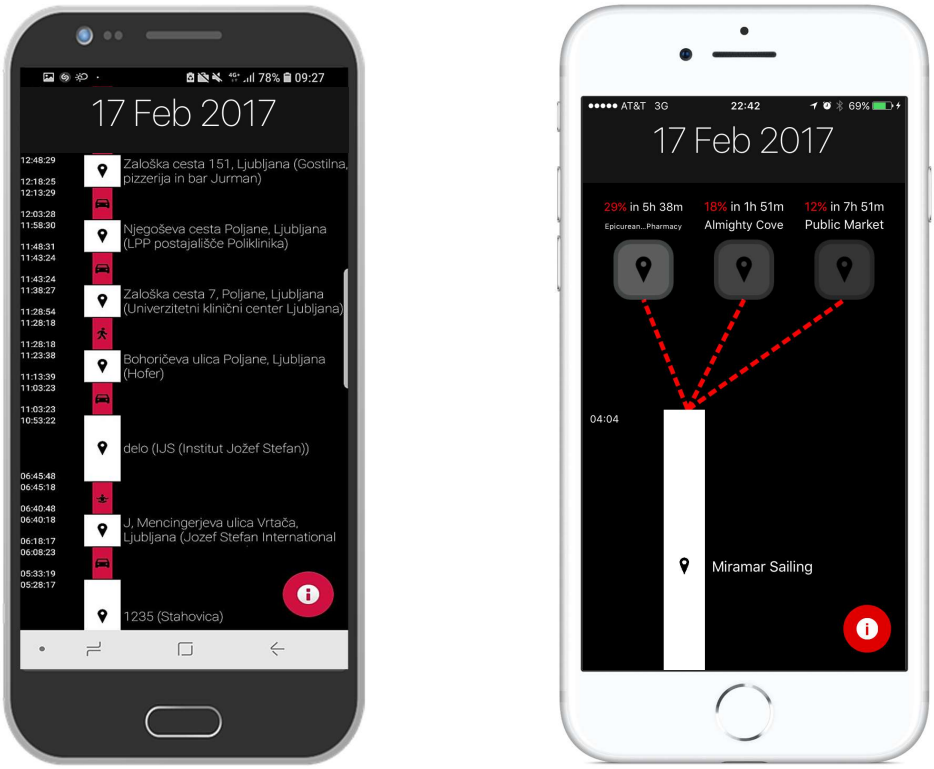


Figure 2.22: A mobile application (Android – left, and iOS – right) was developed for easy data acquisition. The application also visualizes a historical log of users' locations and trips and enables editing. In addition, the top three next location predictions are displayed.

Chapter 3

Time Series Regression Modeling

With the evolution of digitization and data collection, the need for extracting useful information from a vast amount of time series data has grown significantly in the last years. The challenges that come with big data are usually described with the V's of big data: *Velocity, Volume, Value, Variety, Veracity, Variability, Validity, Volatility, etc.* Due to these properties, traditional data mining techniques and methods are continuously facing new challenges and the need for new streaming frameworks is increasing [42]. However, stream modeling has its drawbacks and challenges, and is not always the best solution for all use cases. Roughly we know three (methodologically) different modeling approaches for tackling time series regression problems:

- **Offline models:** traditional data mining methods require the data to be first stored in some sort of a database. Then the entire dataset is initially presented to the modelling algorithm, which can access the data in any order and process it offline. The algorithm produces result in the form of answers, or in the form of models that can be used with a real-time data stream. Such models are called offline models. Even though the model can process streaming data [43], the main drawback of such models is that the model is learned once and is not adapting to promptly changing concepts and data distribution due to the dynamic nature of data streams – a property known as a concept drift [44].
- **Batch models**: tries to overcome previously described problem of concept drift by separating data into several smaller batches (usually by introducing sliding windows), with which they gradually rebuild and update models. The problem is that in order to update the model, the algorithm usually still make several pass over the stored data, but the data stream is infinite and the data generates with high rates, so it is impossible to store it. There is also a problem of learning efficiency, since the model has to pass over the entire dataset when updating its internal model parameters [45]. Therefore this method is not suitable for a large amount of fast streaming applications.
- **Online models**: An ever growing volume and velocity of streams dictates the development of methods that analyse and process data in multidimensional, multi-level, single pass and online manner [46]. The main advantage of online algorithms or incremental algorithms is that the algorithm is capable of inducing a model from the bottom-up approach; each pass of data, from the data stream, triggers the model to incrementally update itself without the need of reloading any previously seen data [47]. These algorithms have limited memory available to them (usually in RAM) and also limited processing time per item, which makes them ideal for processing high velocity data streams at high rates.

While the offline modeling approach is very well researched and commonly used in various practical use cases over the last years, there is still a lack of general methodologies and frameworks which can support a relatively complete understanding of the field, to help developing online models. Some of the main challenges, when developing and deploying truly streaming real-world solutions are: (1) Processing heterogeneous streaming data with different sampling intervals, different acquisition times, dynamic/static in an online fashion; (2) The need of industrial stream pre-processing solutions (data merging and feature engineering, stream aggregates); (3) Many different streaming algorithm implementations, including the most state-of-the-art are academic projects (MOA [48]) which are hard to use on an industrial level, whereas industrial streaming solutions do not offer the latest algorithms (e.g. streaming random forest). In the next chapters, we introduce a general framework for the development of online streaming models, using a novel data analytics platform for processing large-scale real-time streams [49], and apply it on a real-world use case of load forecasting in smart grids.

Load forecasting in smart grids

The electric power system is often described as the most complex system devised by humans [50]. One of the most important aspects of the operation of a power system is the fact that the system response closely follows the load requirements. Due to the significant load fluctuations in the various time periods of each day, it is imperative for the system operator to be aware of the demand that will be expected in the next few hours so that appropriate planning can be performed. This on-demand power generation makes the ability to generate accurate forecast of electricity load requirement one of the most important aspects of effective management. If in the past, experienced system operators were able to predict load requirements, now dynamic complexity of load, expansion of renewable sources, proactive devices, stricter power quality requirements and deregulation are mandating the development of advance load forecasting tools that found the base line to actively manage load and grid.

Load forecasting is an important component in a smart grid environment where almost all involved entities need to perform some sort of forecasts on a continuous basis in order to cope with the increased variable power generation and demand.

- **Generation operators:** mainly use long-term load forecasts for planning purposes such as constructing new power stations and expansion of the electric utility. Short-term load forecasts provide input data for load flow studies and contingency analysis. These are used to calculate the generation requirements.

- **Transmission operators:** Short-term load forecasts are the cornerstone of transmission grid operation, which helps operators estimate future demand for the day-day on-demand operations of the power system. Forecasts of renewable sources power generations are also very important for transmission operators, since they have to optimize the generation of renewable sources according to generation of non-renewable sources.

- **Distribution operators:** use medium-term forecasts for scheduling maintenance and minor infrastructure adjustment. Short-term forecasts are also used for load flow analysis, and power system security studies. Load forecasts are also used in network studies, where operators prepare a list of corrective actions for different types of expected faults. Not only accurate forecasts, also customer segmentation helps network managers to have insight in to the type of customers they have to

supply. These studies are used as a support for network operations and long-term planning.

- **Service providers:** use price forecasts, which include also load forecasts and renewable sources generation forecasts as an input. Mid-term and short-term forecasts are used to schedule adequate energy transactions and prepare long-term bidding strategies.
- **Market operators:** due to the involvement of an increased number of players in the deregulated energy market, load forecasting has become a significant component of energy brokerage systems. Since market prices are also correlated with the generation of renewable sources, such forecasts are also taken into account in decision making.

Even though developing a model for each of them is a very different problem, containing its own stylized features from different data sources, the general forecasting methodology is similar. For this reason we present and propose the general framework for the development of online streaming models in the following chapters.

3.1 Related Work

With a growing number of systems that generate a massive amount of streaming data, the need for real-time stream processing and online analysis is becoming more important. In this section, we examine the state-of-the-art work published in the field most related to fusion of heterogeneous data streams and incremental online modeling. In the second part, we also provide a literature review of work related to electricity load forecasting using probabilistic approaches.

3.1.1 Online processing of heterogeneous streaming data

To achieve the full analytical potential of the data from different sources in the era of the internet of things, the interconnection between those data sources is necessary. By definition, those sources are heterogeneous and their integration is not a trivial task. The system needs to be autonomous and resistant to various situations, including missing, delayed or wrong data. Dealing with non-aligned timestamps from various data sources and fusing them to a common temporal denominator is one of the most important tasks of data a fusion unit. Doing this in real time in an online streaming manner is a challenge that not many researchers have yet addressed. Most of the published work expects all data sources to be temporally aligned, available at the same desired timestamps, and already temporally ordered, which is clearly not the case in real-world scenarios when using multiple IoT data sources. Very few contributions discuss handling delayed and out of order measurements [51], [52]. Or in many cases, systems incorporate some sort of domain knowledge into the pre-processing and fusion processes, by which the approach loses its generalization potential [53], [54]. Our proposed methodology is designed to be very generalizable and is completely domain agnostic.

Literature suggests several approaches for incremental learning with each new measurements. For example, incremental online linear regression [55], density-based clustering algorithms [56], perceptron-based algorithms [57], [58] and incremental Support Vector Machines [59], [60] which incrementally transforms the hyperplane with each new data. Recently, a lot of effort in the literature has also been dedicated to the incremental learning approaches in the deep learning domain [61]. The problem with some of these complex methods is that the algorithm requires a pass through the data with each increment, reducing calculation speed and making it impractical for application with high volume streaming

data. Therefore, some solutions use probability laws to construct an approximation of a batch model. An example of such solutions are Hoeffding trees [62], which use Hoeffding bounds [63] to estimate the split criterion, used when constructing decision trees. Several improvements and extensions of the Hoeffding trees algorithm has been proposed over the recent years (such as FIMT-DD [64] and iSOUP-Tree [65]), where it seems like the most popular is still the Very Fast Decision Tree (VDFT) [62], and its alternative Extremely Fast Decision Tree (EFDT) [66], which is able to learn faster and converges to a batch-like decision tree form. Additional improvement (in speed) was achieved also with Vertical Hoeffding Trees (VHT) [67], which is the first distributed streaming algorithm for decision trees. With our proposed methodology, the output of the pre-processing and fusion module is a single feature vector (fused from different data sources), therefore any incremental machine learning algorithms can be used.

3.1.2 Streaming architectures and frameworks

A general architecture for processing real time web-scale data, called Lambda, is proposed by Marz et al. [61], [68]. The system consists of three layers: the batch layer (Hadoop), serving layer, speed layer (Storm, NoSQL databases). The main features of the system are: simplicity, speed, scalability and ability for debugging. Despite the generic approach and the mention of streaming technologies, authors do not state any details on data fusion implementation.

Bouguelia et al. [69] presented a domain-free framework extracting high-level conceptual knowledge from multiple streams of data. It is based on aggregating time serieses over time using moving average aggregates and clustering methods for constructing fixed feature vectors. The advantage of our proposed method is that it is also able to fuse heterogeneous data that are non-uniformly sampled and delayed.

Recently, challenges of processing large amounts of streaming data from IoT sensors were addressed in the work by Tu et al. [70]. They extend the basic windowing algorithm for real-time integration (i.e. ISDI framework) in a way that is able to deal with the non-uniformly sampled data from different data sources (time alignment issue).

As it can be observed from the literature, the main limitation of existing work is still a general (domain agnostic) stream mining framework that can be used in sensor network-based applications. Many stream mining frameworks were mostly focused on solving descriptive data mining problems in the web analytics domain (rule extraction, outlier detection, clustering), rather than predictive time series problems (forecasting future values). Although, the number of published research with proposed stream mining frameworks for processing real-time sensor data is notable over the past few years. And many of these proposed frameworks and architectures already tackle the problem how to process massive amounts of real-time streaming data. Not many cover the challenges of real-world challenges such as online fusion of non-uniformly sampled heterogeneous data sources, or fusing delayed data streams, real-time data streams and data streams from the future (e.g. weather forecast), which we address and cover with our proposed approach.

3.1.3 Load forecasting in smart grids

We can systematically observe that researchers [71]–[73] most commonly divide forecasting approaches in different groups, with respect to forecasting horizon: long-term, medium-term, short-term and lately also very short-term forecasts. In this work, we focus on short-term load forecasting, but we also mention the other two groups for the sake of completeness.

- **Long-Term Load Forecasting (LTLF)** covers forecasting periods up to 20 years ahead. These studies are generally used for planning purposes, such as constructing new power stations and expansion planning [73]. Typical input data sources for such analysis are *historical data consumption* data and socio-economical indicators such as *population growth*, *industrial expansion*, *gross domestic product*, *past annual energy consumption*, etc. [74]. The most commonly used modelling techniques are statistical parametric methods [75] as well as non-linear artificial intelligence approaches [74], [76].

- **Medium-Term Load Forecasting (MTLF)** covers forecasting periods from a few weeks to one year ahead. Commonly used for scheduling maintenance, fuel supply planning and congestion management, therefore improving overall system efficiency of the transmission grid. Typical input data sources are *various seasonal patterns (holidays, events, weather)*, *economic* and *demographic factors* [73]. Similar as with LTLF, the two most commonly used modeling groups are regression-based methods [77]–[80] and AI-based approaches (mostly ANN and SVM) [81]–[84].

- **Short-Term Load Forecasting (STLF)** covers forecasting horizons from one hour to one week into the future. The biggest number of research papers related to electricity load forecasting fall into this group, as it is the most convenient use case for data-driven models (in terms of prediction horizons and data availability). STLF applications are typically used for day-to-day operation, scheduling energy transactions, contingency analysis, load flow analysis, etc. Usually three main groups of input data are used: *seasonal input variables* (e.g. load variations caused by heating units [85]–[87]), *weather forecast variables* (temperature is the most important one [88]–[90]), *historical load variables* (which describe short-term dynamics [86]) and *static data* (to incorporate exceptions such as holidays and social events [86]). Modelling approaches used for STLF are discussed in detail below.

- **Very Short-Term Load Forecasting (VSTLF)** covers forecasting periods from seconds to one hour into the future [71]. This group is very recent due to recent advances in sensor technology (IoT) and data availability. Most commonly used for operation optimization of micro systems. Due to the very short-term prediction horizons, low latency and fast computation times are of essential importance. Due to this reason, the most common input data sources are various data sources from the concerned modeled system that can be obtained almost immediately and as often as possible (different internal sensors, e.g. wide area management systems (WAMS), accelerometer, temperature, etc.). External data sources, such as weather, calendar and social events, are not that important in case of such short-term prediction horizons, as the influence of these factors is already incorporated into the latest measured data. This field is still relatively new, but some research has been done using ARIMA [91], ensemble of kernel-based Gaussian processes [92] and wavelet neural networks [93].

Evidently, a vast amount of methods and techniques for forecasting on power grids have been developed over the past years and yet no single modelling approach clearly sticks out. The main reason for such a large amount of different methods is that each system strongly varies in different characteristics, such as geographical properties, meteorological factors, network topology, type of related external data source, etc. Additionally, availability of data from the emerging field of sensor technologies and computer intelligence is enabling the development of new sophisticated models. There are several surveys that provide a good overview of a variety of techniques used in the demand prediction field [71]–[73], [81],

[85], [86], [94], [95]. Most of the authors group the approaches into two main classes: *classic statistical approach* and *artificial intelligence approaches*. For the sake of completeness, we add two additional groups: *hybrid*, and *naive approaches*.

1. **Classic statistical (CS) approaches**: are traditionally used methods such as autoregressive time series models and moving averages (ARMA), regression-based techniques and Kalman filtering.

    - *Autoregressive time series* are among the oldest methods used for load forecasting. A typical method is autoregressive moving average (ARMA), where historical data of prediction variable (load) is used to recognize short-term trends that are used for estimating short-term predictions. Different variations of ARMA models, such as ARMAX or ARIMAX, include also seasonal information, which typically improve predictions. External data sources, such as weather, are usually not included in these methods [75].
    - *Regression-based models* are another very common group in classical model approaches. These methods are used to model relations between different input data and the target. Additional data sources, such as weather, can be included. The load is represented as a linear combination of variables related to load, such as the weather factors, day type, customer class. This approach offers a lightweight solution that often offers good results. This proved to be the case in a competition of Electric Power Research, where Ramanathan et al. [96] proposed an effective regression model (24 models – for each hour in a day), which outperformed other methods. Refinements of regression model were proposed by [97], by including weather information, holiday information and outlier removal, which improved the overall performance of the model.

2. **Artificial intelligence (AI) approaches**: are becoming more and more interesting in the last few years, mainly because of computational advances and availability of large datasets, which is crucial for these methods. The main advantage of such methods is that complex relations between the data sources and the target can be modelled. CI approaches contain methods such as deep neural networks, support vector machines, expert systems, evolutionary systems.

    - *Support Vector Machines (SVM)* were introduced relatively recently to the field of STLF, but during the last few years there has been a lot of work reported. It is a non-linear kernel-based approach (RBF kernel is very commonly used in STLF), which proves to be very resistant to over-fitting. Some authors even state that it has been proven that SVR provide more stable predictions than basic NN [98]. SVM has been proven to be very suitable also for mid-term load forecasting use cases, by winning the competition of EUNITE network [99].
    - *Artificial Neural Networks (ANN)* are most often applied from this relatively new group of approaches. Several subtypes of ANNs exists, but feed forward (also called multilayer perceptron) are the most frequent. From the reviewed literature we can also observe that frequently only one hidden layer is used and produces good results. Several authors report best results for STLF by using ANN compared to other methods [74], [76], [100].
    - *Deep Neural Networks* have recently been a hot topic in many fields such as in STLF. In many cases authors report improvements over benchmark methods (e.g. ARIMA, NN, SVR)[101].

3. **Hybrid approaches:** with the vast amount of methods already at hand, quite a lot of researchers have studied combining different methods for potentially better accuracy and robustness to noise and random errors. Generally, these hybrid approaches combine two or more different approaches in order to take advantage of specific methods' benefits and overcome their drawbacks. Frequently, combinations of classical approaches and AI approaches can be observed. Most popular combinations are fuzzy logic methods with neural networks (also called ANFIS) or wits support vector machines [72], [85]. Several methods exists for combining outputs of different models in order to generate ensemble forecasts. These approaches are especially used in numerical weather forecasting [102]. Usually simple strategies define how different outputs will be combined in the form of fuzzy logic rules [98]. An example of simple strategies are mean of predictions, median of predictions, switching among predictions using the one with the smallest error. Another popular method for combining different prediction outputs is by using them as an input in another model (usually ANN) which produces the final forecast [103]. Generally, results show that by using a hybrid approach we get better predictions than by using individual models alone [98].

4. **Naive approaches:** one group of methods that is usually missing in reviewed work are naive methods. These are considered as simple, unsophisticated forecast approach, where no forecast model is actually built. They are computationally non-demanding, therefore they are very fast, robust and easy to maintain. An example of the naive method are "instantaneous prediction" where the last measured value is used for prediction, or "historical value prediction", where the similar measure to forecasted measure from the historical database is used for prediction (for example, previous day, or previous week). Another example would be different variations of "exponential moving averages" which encapsulate the latest trend very well and are commonly used as a simple, yet effective, approach in very short-term load forecasting [91]. Despite the simplicity of such naive methods, they are still widely used in various fields, due to their fast computational capabilities and transparency, which can also often generate surprisingly good predictions. These methods are usually used as baseline methods, for comparisons with other more complex methods. If a newly developed method is not performing better than some naive method, then there is no sense in using a more complex method. When comparing different methods and deciding which one is the most appropriate to use, one should not take only error metrics in to consideration, as there is a thin line between the model accuracy, simplicity and suitability, where all three should be considered [104].

It is very clear that numerous different techniques were reported over the last decades and that artificial intelligence approaches are becoming more and more popular for electricity load forecasting. However, it is also obvious that there is no universal best technique that outperforms all others. The selection of the most appropriate technique is determined by the available data, data properties, business needs, prediction horizon, etc. Another challenge that we can observe from the reviewed literature is the lack of a general stream mining framework that can be used in sensor network-based applications. Most of the applications in the sensor network domain are still designed by using offline batch methods, with ability to retrain and update models on request, but are not based on principles of incremental learning and lightweight processing. Some stream mining frameworks for sensor networks were also proposed in the past few years, but were not broadly adopted by other researchers, as they were all very specific to their own use cases. Also, the described stream mining frameworks were mostly focused on solving descriptive data mining problems (rule

extraction, outlier detection, clustering), rather than predictive time series problems (forecasting future values). In this work, we are proposing a general stream mining framework specialized for real-time large-scale sensor streams. Although the methodology should work with any time series regression modeling technique, it was designed for streaming use cases, therefore it works best with data streaming techniques that enable incremental learning (e.g. NN, recursive (incremental) linear regression, Stochastic Gradient Decent regressor, Mondrian Forest, etc.). In Chapter 3.3, we apply the proposed methodology on the real-world of STLF use case and extensively evaluate results of learned models, using different modeling techniques and input feature sets.

3.2 Methodology

Our proposed methodology for stream modeling development is based on a well-established cross-industry data mining methodology called CRISP-DM [105], originally designed for traditional offline use cases. Figure 3.1 visualises the entire process, showing the relationship between the different phases of the data mining process. In the diagram, data is put into the center, which perfectly symbolizes that everything depends and moves around the data that is available. The outer circle in the diagram symbolizes the cyclic nature of data mining itself, which is that a data mining process continues after a solution has been deployed. The lessons learned during the process can trigger new, often more focused business questions and subsequent data mining processes will benefit from the experiences of the previous ones. The process (described by CRISP) breaks down into six major phases: *Business understanding, Data understanding, Modeling, Evaluation, Deployment.*

Even though the described CRISP process was initially designed to describe classical off-line data mining approaches, it can still be used to design online approaches by slightly generalizing the schema, since the modelling stages stay the same, only the methods are adjusted to meet the online streaming demands. For this purpose, we have grouped the described six stages of the CRISP data mining process into three higher level stages where the steps stay the same for both offline and online modelling, only the methods are adjusted for streaming systems: *Exploratory analysis*, *Modelling* and *Deployment* step (see Figure 3.1). In the next sections, we describe each step in detail.

3.2.1 Exploratory data analysis

In order to develop a successful model, one must incorporate as much domain knowledge as possible. To do so efficiently, it is crucial to understand the problem (business) well, as well as the data. We achieve the former by collaborating with business stakeholders and the latter by exploratory data analysis (EDA). While the main purpose of exploratory data analysis is to understand a dataset and summarize their main characteristics without making any assumptions about its contents.

In general, EDA is the practice of using visual and statistical methods to describe the data (e.g. normal behaviour, anomalies, patterns, outliers, correlations, etc.). The outcomes provide a context needed for the development of an appropriate model for the specific problem and to correctly interpret its results. There are no hard-and-fast rules on how to approach it, as it highly depends on the type of problem and the data available. Some of the most common tasks in EDA that help us gain insight into our data are: *analysing the underlying structure of the data, detecting missing and faulty data, spotting anomalies and outliers, identifying patterns, identifying important features, analysing feature correlations, hypothesis testing and validating assumptions, etc.*

If in the past, the dataset owners were aware of the data and their properties, nowadays,

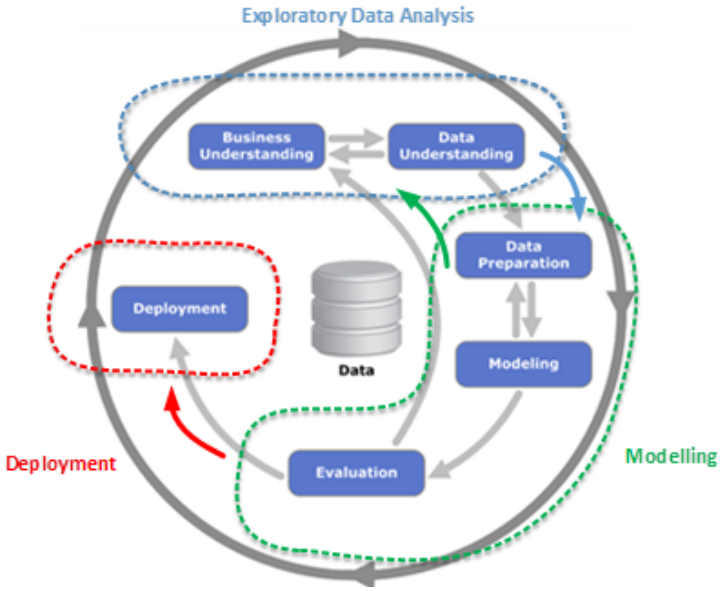


Figure 3.1: CRISP-DM methodology [105] steps grouped in to three higher level stages (i.e. *Exploratory analysis*, *Modelling* and *Deployment* step).

in the era of big data, companies produce datasets (i.e. logs), for which they do not have the intuition. By performing extensive EDA researcher we gain insight (that might be new also for the data owners) and build important intuition for the data required during the entire modelling process. Preprocessing is the first stage of the modelling process. It is important to gain insight of the data so that the researcher knows how to pre-process the data correctly and how to audit them (missing data, corrupt data, outliers, etc.). When dealing with heterogeneous data sources, data fusion is another important step if we want to take advantage of the knowledge from all this different sources. Also data alignment is one of the biggest issues, since there are so many different types of data sources that produce so many different outputs (dynamic and static). Data must be transformed from each source's local frame into a common frame. There are a lot more issues concerning data fusion, such as data dimensionality, data correlation, data modality. The task of data preprocessing becomes even harder when we are building an online streaming application. Streaming modeling methods expect data to arrive in a timely fashion. When dealing with a multivariate data stream many things do not match: data stream frequency is different, data is updated using different protocols (some data is coming on-line, other sources send data in many small batches). Known methods that are usually used in the data fusion process, such as data interpolation and extrapolation, became a lot harder problems with streaming data. Extensive EDA helps to tackle all of these problems during the modeling phase.

When developing the model, the results can only be as good as the quality of input data, therefore the pre-processing step is often considered as one of the crucial steps in the modelling process. EDA also provides crucial domain knowledge insights, useful during the feature engineering and modeling stage. By building the intuition for the data, we can also better audit the final results and quickly discover irregularities in the output data or mistakes in the implementation of a model. Based on our proposed methodology, EDA step also has to include some of the preliminary offline modeling steps (data preprocessing, feature engineering, model selection and evaluation). Finally, the outcomes of EDA should give us a very clear understanding on how to further develop a streaming model (described in the next section).

3.2.2 Stream modeling framework

For stream modeling we proposed a generic framework [106] with three main building blocks; *Pre-processing* module is responsible for data enrichment of a particular stream, *Fusion* component is dedicated to merge and align all the streams on the same time interval, producing a single feature vector ready for the *Modeling* component which contains online modeling algorithms (see Figure 3.2). Building block are described in detail in the following subsections.

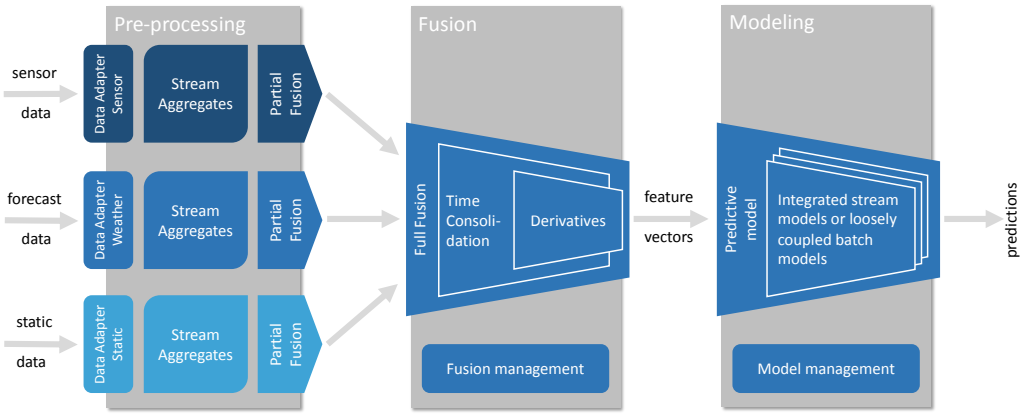


Figure 3.2: Streaming framework includes three main components; (1) *Pre-processing*, (2) *Fusion* component and (3) *Modeling* component.

3.2.2.1 Pre-processing component

The module consists of adapters for different data streams, followed by the custom designed cleaning operators, such as outlier detectors, corrupt data handling, data normalization. Cleaning operators are derived from the exploratory analysis, which is the key for effective data cleaning. Resampler transforms the stream to the common time interval and also handles with possible missing data (down-sampling or up-sampling). Various stream aggregates are then used to enrich the data source. Stream aggregates are data processing components that consume one or more data streams as input and produce an output stream. Data enrichment starts from an initial set of measured data and produces derived values (features) intended to be more informative to the predictive models (and sometimes also to humans). We obtain higher level features with historical values and autoregressive derivatives (e.g. different lengths of moving averages, variances, average mins, maxs, etc.), which results in partial feature vectors. The knowledge which features to extract is again derived from the exploratory data analysis. The output of this module are multiple partial feature vectors, produced by Partial Fusion component (see Figure 3.2).

3.2.2.2 Fusion component

Fusion component is responsible for merging all partial feature vectors from different data sources and resampling all streams to the common master time interval. Potentially different timestamps over different data sources are compensated by down-sampling or interpolating. Our solution (which contains QMiner [49]) supports different interpolation methods; *linear interpolation*, *last value interpolation* and *time tick interpolation*. These affect the behaviour of the system, as the linear method requires two points to interpolate, and consequently introduce a temporal delay, until the second point in time is known.

While the last value and time tick interpolation method only requires one point (in a streaming scenario this is the last received measurement), avoiding the delay, but sacrifices approximation accuracy, since the last point can only be repeated. At this stage, additional derivatives (across different data streams) are also calculated from partial feature vectors, such as the difference between the latest and daily average electricity consumption. The output of this component is a single feature vector, containing measurements and their derivatives from all available data sources ready to be used by the analytical modeling module.

3.2.2.3 Modeling component

At this point, heterogeneous data streams are transformed into a structure which can be used by the modelling algorithms. Our framework supports classical offline, batch and online modeling approaches (also called incremental algorithms). Here we describe the most interesting one, the online modeling approach, which usually consists of three main steps: *updating phase*, *learning phase* and *evaluation phase* (See Figure 3.3).

- **Updating phase:** in our proposed approach, we introduce a component called "Window buffer", which is the same size as the largest prediction horizon window. When the new feature vector arrives, the latest measurement value from this vector is used as target for the model, but the training feature vector has to be retrieved from the buffer, based on the prediction horizon information. We propose a decentralized modeling approach, where each prediction horizon is modelled separately with its own model. The buffer (which is sometimes also a hash map) is actually holding the pointers to the model instances responsible for a certain horizon. Based on the prediction horizon, we select the appropriate modeling instance, which also contains the appropriate historical feature vector, used for updating the model, with the latest measurement used as a target.

- **Learning phase:** uses the latest feature vector provided by the Fusion component for making predictions for various horizons by using different model instances. As already mentioned, we use different localized models for different prediction horizons (e.g. from 1 h into the future, to 24 h into the future). The predictions are also stored in a "Window buffer", so that we can use them for the evaluation step.

- **Evaluation phase:** we use the latest measurement and buffered prediction from the appropriate modeling instance (based on the desired prediction horizon) to calculate different evaluation metrics, which helps us evaluate the performance of the regression model (see Table A.3 in Section A.2.2 of Appendix A). Similar as with incremental learning algorithms, some of the error metrics can also be calculated in an online/incremental fashion. The most common are mean absolute error (MAE), mean absolute percentage error (MAPE) and coefficient of determination (R2).

3.2.3 Model deployment

The proposed framework is designed to be used in lambda and similar big data architectures [68]. Below we present the integration of two common scenarios of different complexities; (1) *cloud solution* and (2) *edge solution*.

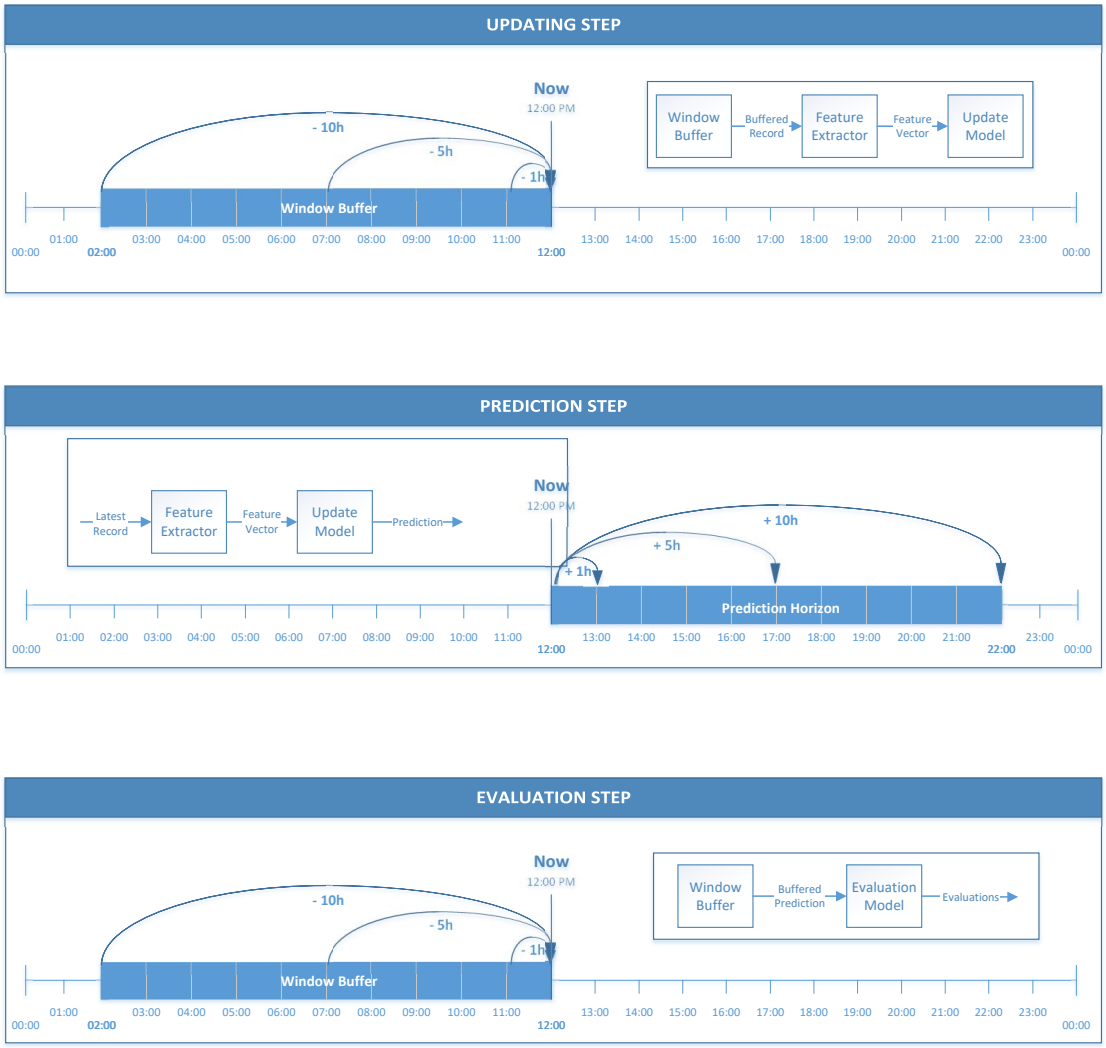


Figure 3.3: Three usual steps of an incremental modeling algorithm; *updating step* where the current measurement and historical feature vectors are used, *prediction step* where the current feature vector is used for prediction, and *evaluation step* where historical predictions are evaluated based on the current measurement.

3.2.3.1 Cloud solution

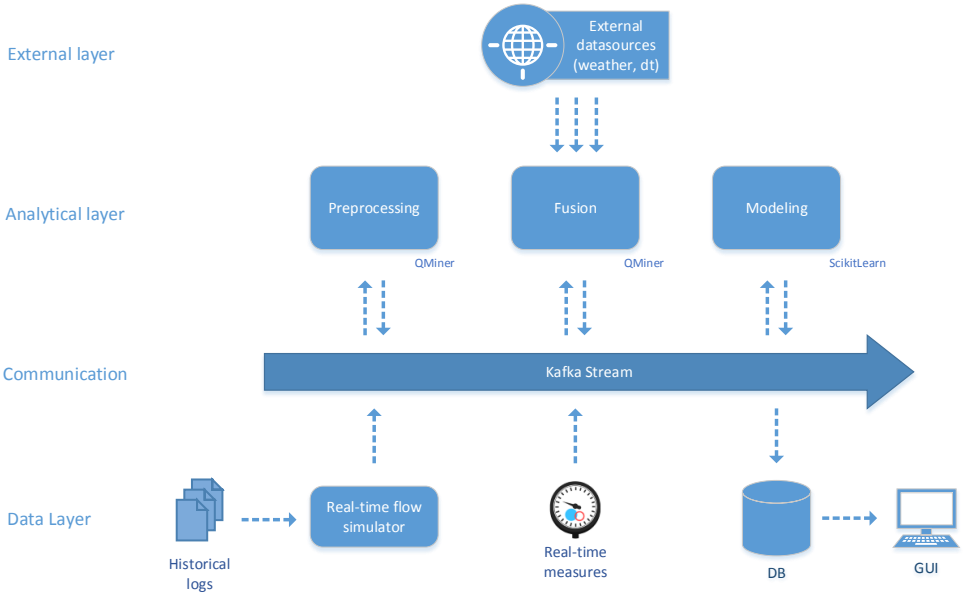
The most common implementation of the proposed framework is the cloud solution, where the main solution is deployed on a remote server. As shown in Figure 3.4a, the solution consists of four main layers; *Analytical layer*, *Communication layer*, *Data layer* and *External layer*. Analytical layer represents the same three building blocks presented in the previous subchapter (Section 3.2.2). The preprocessing and Fusion component are using the QMiner analytical platform [49], which was designed especially for processing a large amount of heterogeneous streaming data in real time. Since the framework supports offline and online approaches in the modeling component, we usually use Scikit-learn library [107] for offline modeling and MOA [48] and QMiners [49] incremental modeling algorithms for online modeling. The communication between the components is enabled via message queue system within the Communication layer. In our solution we use Apache Kafka which is a distributed streaming platform designed especially for building real-time data pipelines. By loosely coupling our components, the system is a horizontally scalable and fault-tolerant infrastructure.

Data are obtained by the framework via the Data layer. If we have historical data available for pre-training the models which can substantially speed up the modelling process, we can simulate the dataflow using a real-time simulator (which can be sped up). After the model has been trained, we can start listening for new actual real-time measurements. The system can also send data back to the data layer (via communication layer) such as intermediate results from a certain module (e.g. partial feature vectors from the pre-processing component) or the final predictions from the modeling component that we want to store to some external database, from where it can be accessed by the GUI client. Additionally we also introduce the External data layer, which is used for obtaining additional external data sources from the internet (such as weather etc.), which can improve the models' predictive performance. We have successfully used this solution in scenarios such as traffic flow prediction [108], [109] and short-term load forecasting in the electricity distribution grid (which we present in detail in the following chapters) [110], [111].

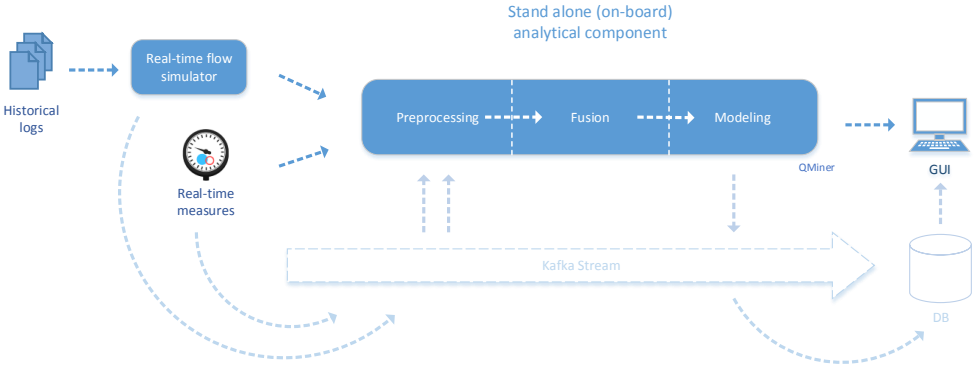
3.2.3.2 Edge solution

Due to the rising popularity of smartphones and low computing power IoT devices, which were originally meant for data acquisition, the rise of demand for stand-alone infrastructures able to function on devices with limited processing power has been detected. Such systems are called edge solutions, since they are deployed on the same unit (or very near) where the measurements are taken. There are several benefits of such an approach, such as parallelization of work, faster response times, cost-effectiveness, no need for internet connection, security, etc. But the limitation is often the low computing power on such systems, but the development and performance of these devices is increasing each day.

The solution presented in Figure 3.4b was adjusted to the constraints of the edge computing use case. The main difference from the previously presented infrastructure is that the three modeling workflow components are now tightly coupled together. This makes the pipeline processing much faster and eliminates the need for a data layer with a messaging queue system since the data flow is completely data driven between the tightly coupled components. Still, when needed, we have also used a lightweight message queue solution based on the MQTT protocol. Since these types of solutions are usually used in cases with no internet connection, there are also less data sources that have to be merged, therefore the complexity of the system is greatly reduced, which results in greater reliability and high latency. We have successfully used this solution for estimating short-term electricity demand in electric trains.



(a) Cloud infrastructure solution contains four main layers; *analytical layer* (includes the streaming framework presented in Section 3.2.2), *communication layer* (distributed message queue system), *data layer* and *external layer* (providing the data inputs and outputs).



(b) In the edge solution, the architecture is simplified. The modeling components are tightly coupled together, which eliminates the need for a data layer. There are usually a lot less data sources available, therefore the complexity of the system is greatly reduced, which results in higher responsive times and greater reliability.

Figure 3.4: Integration of two common modeling scenarios of different complexities.

3.3 Short-Term Load Forecasting

We employ the proposed framework on the real-world scenario use case of short-term load forecasting. Load forecasting is an important component to almost all entities involved in the management of a modern electric grid, especially in the deregulated environment with increased variable power generation and demand. Generation companies, transmission companies, distribution owners, service providers, as well as market operators, they all use different forecasts in their decision-making processes and day-to-day operations [95]. Short-term load forecasting is an ongoing research topic for over 50 years, which has become even more interesting in recent years due to the recent availability of large datasets (e.g. AMI meters that were primarily used for billing purposes) and increasing capabilities of AI methods.

3.3.1 Data description

The data acquired for the STLF use case covers the most important groups of input variables, recognized from the related work research in Chapter 3.1. These are: *historical load data*, *seasonal variables*, *meteorological variables* and *additional static data* (see Table 3.1).

For our experiment, energy-related data were provided by the Slovenian Power Distribution Network Operator, Elektro Primorska, during the FP7 EU project Sunseed [110]. Historical dataset contains measurements from 20 smart meters (AMI) from the Kromberk test field, over the period of 4 years (2011-2015). Logs were provided as time series tuples, containing measurements from each smart meter with a timestamp. In addition, these measurements were also used to calculate power flows in the branches positioned on the higher levels of the tree grid architecture, by taking advantage of the grid topology (marked as Load Estimations in Table 3.1). Beside the historical dataset, which is used in exploratory analysis and model selection analysis, we are also getting near real-time data feed, which is used in the production version of the prototype, which is processing the data in an online fashion.

Another important source is weather data, since it is a well-known fact that meteorological conditions have a significant influence on electricity loads [86]. Weather data was acquired from the leading environmental institution in the region, ARSO (Slovenian Environmental Agency)[1]. Several weather parameters were collected for the same time period as load data, from the closest weather station: pressure, temperature, humidity, precipitation accumulation, wind speed, energy radiation, etc. From the related work, it was also recognized that accurate weather forecasts are an important feature. Since ARSO does not provide historical weather forecasts, in this research we used shifted historical weather measurements as estimates of forecasts. We took this approach as a proof of concept in order to analyse the importance of forecast weather information versus the observed weather measurements.

Beside the dynamic data described above, static data was also used. Typically, an example of static data are calendar data such as *"hour of day"*, *"day of week"*, *"month of year"*. Additionally, we include "regular" anomalous dates, for which we are aware of in advance (such as national holidays, start and end of daylight saving times, school holidays, etc.), since it is a known fact that these dates lead to systematic variations in the electric load and can therefore be very useful information for the predictive model.

[1] `http://www.arso.gov.si/`

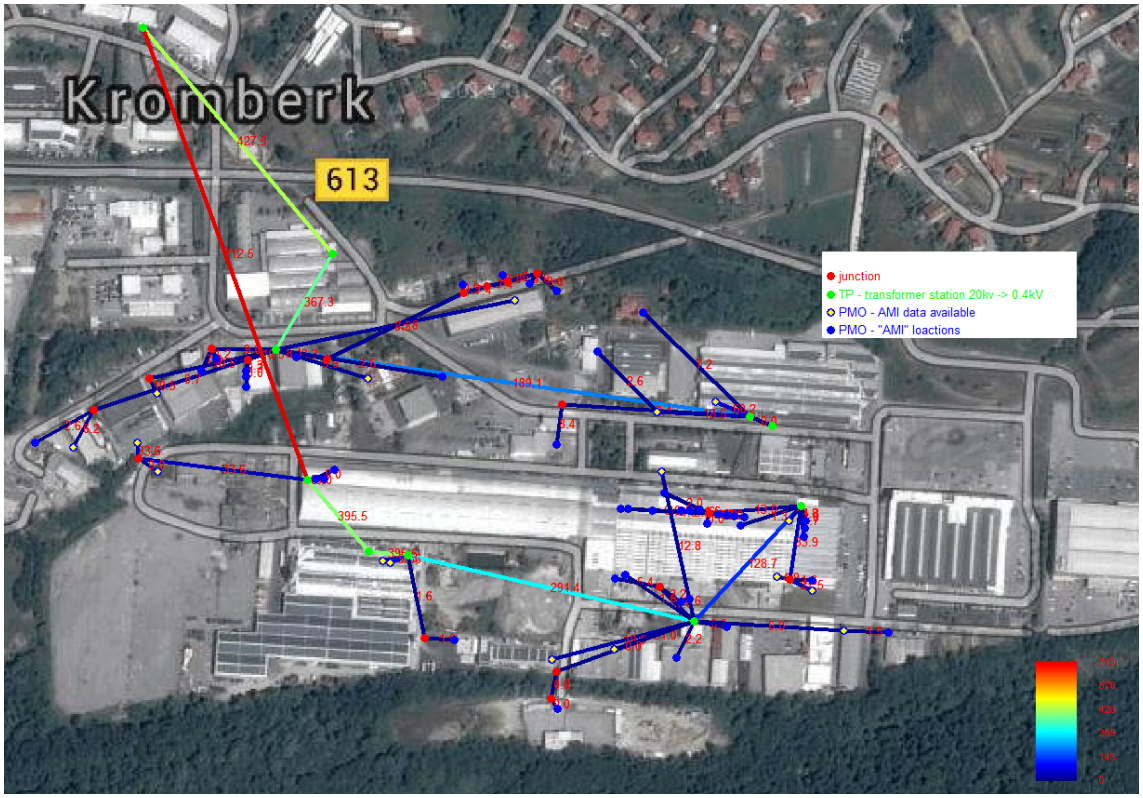


Figure 3.5: Kromberk test field (data from 21 AMI smart meters).

Table 3.1: Data sources used for the STLF use case.

Data Sources	**Time Period**	**Data Frequency**	**Parameters**	**Comments**
Load Measures	1.1.2011-19.2.2015	15 min	20 AMI meters	Industrial sensors from the Kromberk test field
Load Estimations	1.1.2011-19.2.2015	15 min	30 modes	Load estimation (from AMI measures) on 30 different nodes from the Kromberk grid
Weather	1.1.2011-19.2.2015	30 min	weather measurements	Data are obtained from the Slovenian Environment Agency (ARSO)
Weather Forecast	1.1.2011-19.2.2015	30 min	weather forecasts	Shifted observed weather is used to simulate forecasts
Static Data	1.1.2011-19.2.2015	15 min	calendar features	Valuable for periodical data and non-regular days

3.3.2 Exploratory data analysis

We performed an extensive exploratory data analysis of the acquired data in order to understand the underlying structure in the data. This analysis focuses on identifying patterns, outliers and other interesting anomalies. We later used this knowledge to create valuable features, or to detect hidden errors and irregularities in the data, which we have to deal with in order to design better models.

Level of load aggregation has a strong effect on the forecasting performance [112], we include four different measurement points into our analysis, with different average load consumptions at different levels of aggregation. Figure 3.6 shows the mentioned nodes on the map and shows a small time series subset plot of its measurements. The first two nodes (i.e *Kromberk industrijska cesta* and *PMO Upravna stavba*) are measurement nodes from commercial buildings, with mean load values 1.56 kW and 11.89 kW. The third measurement node (i.e. *Kotlarna*) is from a transformer station, where more users are aggregated and the average load is therefore higher (107.44 kW). The fourth measurement node (i.e. *RTP Gorica 110/20kV*) is the main distribution transformer station and it is the highest possible aggregation node on this testbed (with a mean load of 524.98 kW).

Table 3.2: Network nodes from Kromberk used in the evaluation (from 1kW to 500kW).

Node ID	**Type**	**Name**	**Mean** ***[kW]***	**Std** ***[kW]***	**Range** ***[kW]***
id11010024	Prosumer	Kromberk - Industrijska c.	1.56	1.91	8
id14061721	Prosumer	PMO upravna stavba	11.89	7.23	34
id12041992	Transformer	Kotlarna	107.44	38.92	171
id12041022	Feeder	RTP Gorica 110/20 kV	524.98	323.71	1053

As expected, a distinct weekly trend can be observed for several different sensors (see Figure 3.6). Difference in electricity consumption between business days and weekend days is also clearly seen in the mentioned figure. This pattern can also be observed in Figure 3.7, which presents aggregated daily averages separately for winter days (see Figure 3.7a) and summer days (see Figure 3.7b). This figure also shows the daily trends, such as working hours, lunch times, and the end of shift. Daily patterns are clearly different in these two seasons, due to temperature differences and different daylight periods, which also indicates the importance of season information. As expected for the Slovenian geographical region, more electricity consumption is detected during the winter months (due to heating), and less in the summer months (since summers in Slovenia are mild and not a lot of air conditioning systems are installed).

In order to explore the relation between parameters from different data sources and target parameters (load measurements), Pearson correlation coefficient was calculated between features from different data sources and visualised in the form of a heat-map in Figure 3.8. Brighter colours indicate stronger correlations (red is positive correlation, blue is negative correlation – we are interested in both), while lighter colors indicate less correlation. As expected, results show significant correlations between load measures and estimated measures since the estimated measures are aggregates of load measures. We can also observe clear a correlation between power consumption and some weather features, such as *temperature*, *energy radiation* and *humidity*. From the date time group of features, almost all features, except for "Month", show a clear correlation with load measurements. The most probable explanation for this is that load dynamics show a lot shorter cycle (on a daily basis) than the "Month" feature (yearly basis).

Figure 3.8 also reveals correlations which are undesired for some machine learning model (e.g. linear models), these are correlations between input features. For example,

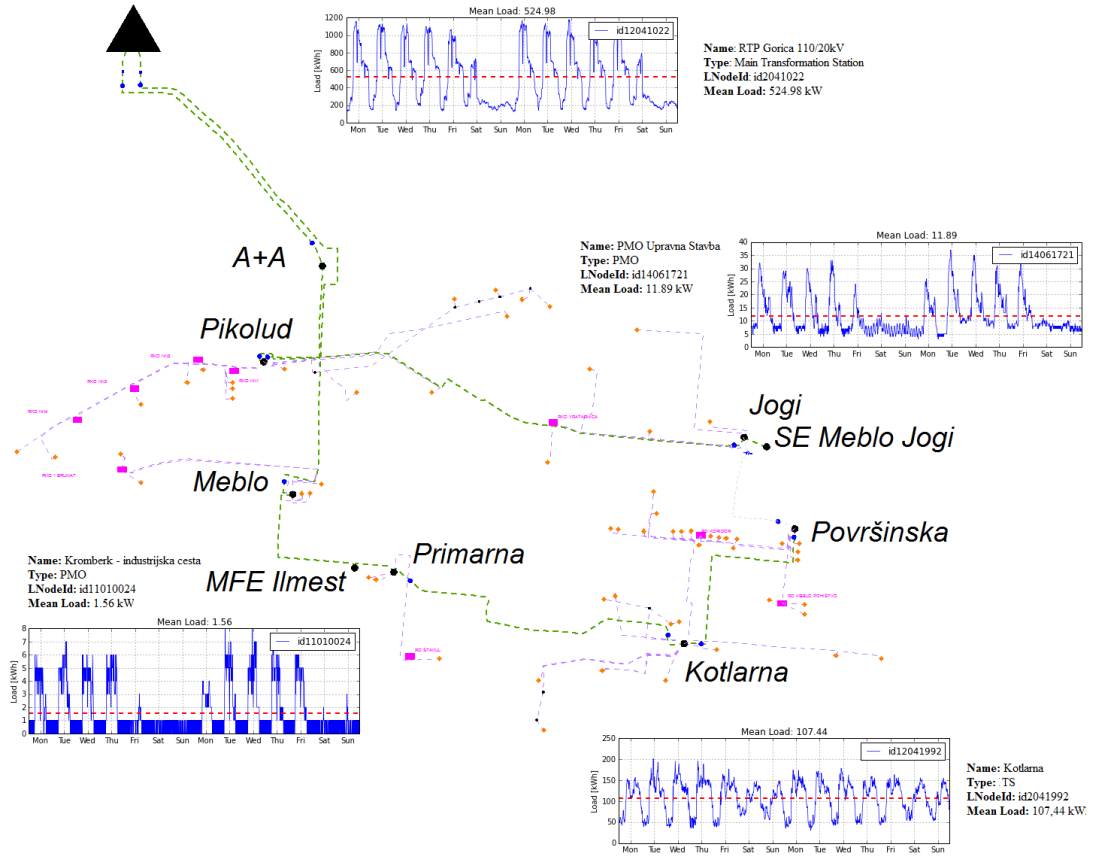
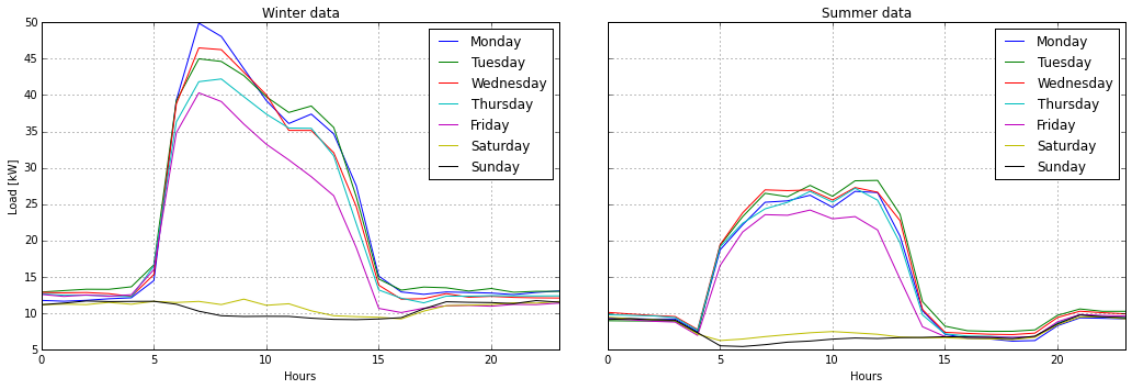


Figure 3.6: Four different evaluation nodes of different levels of aggregation (from 1 kW to 500 kW), from the Kromberk test field.



(a) Winter season

(b) Summer season

Figure 3.7: The figure shows winter and summer daily trends. We can observe the clear difference of the load consumption pattern between these two seasons. The difference between business days and weekend days is also very noticeable (for both seasons).

we can observe a correlation between *Month* and *temperature* features, which are both potential input features for the power consumption model. In theory, input features in the model should be independent between each other in order to avoid confusion and redundant information. In other words, seasonable variable *month of the year* information, and meteorological variable *temperature* can contribute the same information to the load forecasting model twice, and can therefore worsen the prediction when using some methods (Naïve Bayes algorithm is such classic example). Multicolinearity can also yield solutions that are wildly varying and possibly numerically unstable. Similar correlation was observed between seasonable variable *hour of day*, and meteorological variable *radiance* (which gives information about daylight). In order to obtain the best forecasting results, feature engineering and feature selection are the most important parts of the modelling process.

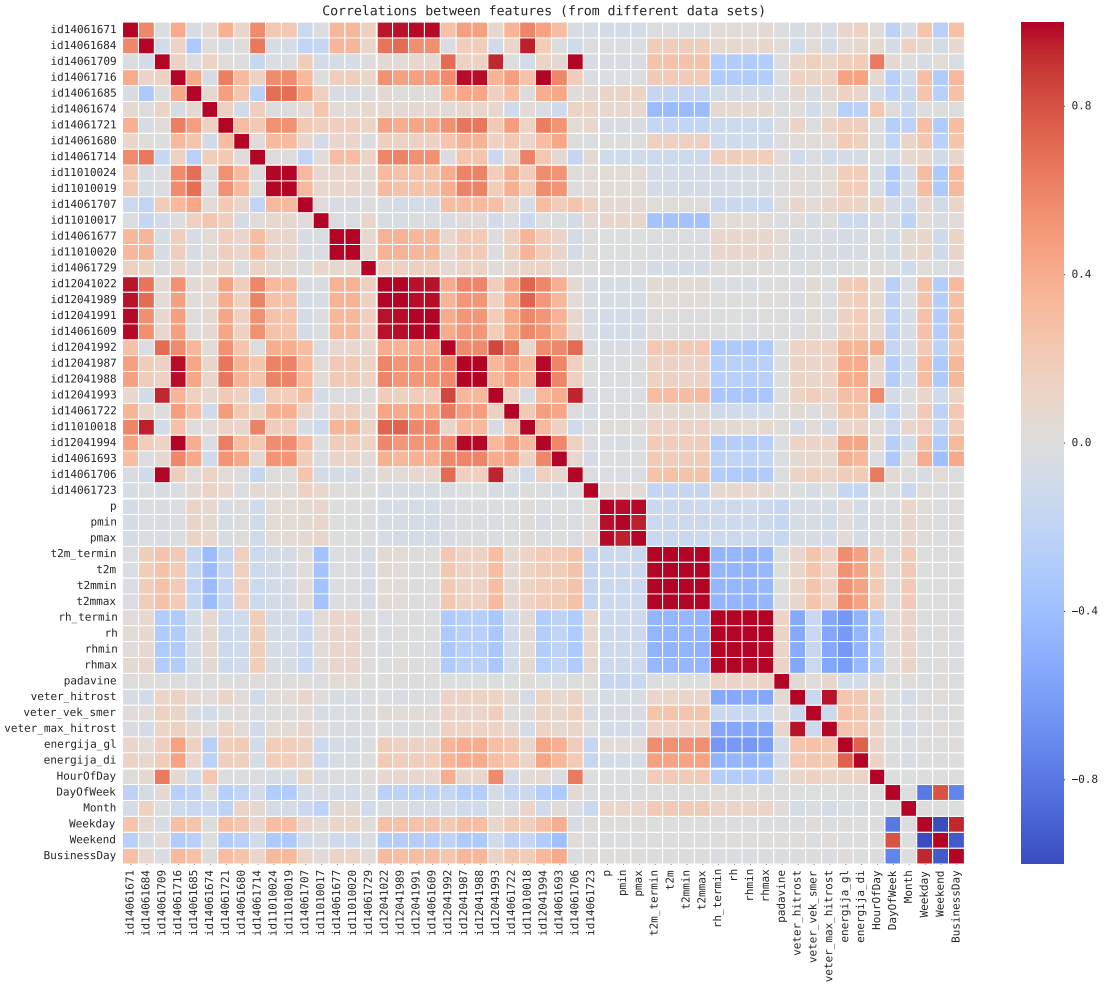


Figure 3.8: Pearson correlation coefficient between input data from obtained data sources.

3.3.3 Feature engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models. For each data source (see Table 3.1) we extracted additional features for which we believed might improve forecasting models performance:

- **Sensor:** One of the most important types of data for STLF modelling are historical load values. Not only do we indisputably need a sufficient amount of historical data for learning the models, they are also used as features as an input in these models. These are features, such as hourly loads for the previous hour, the previous day, and the same day of the previous week. Short-term trends can be computed from the historical data. Updating localized averages of the target variable can be very informative as well [86].

- **Weather:** Meteorological conditions have a significant influence on the electricity load. Variables, such as temperature, wind speed, solar radiation, or rainfall, have been used in related literature [86]. For STLF, temperature is the most important factor [88]. It is quite often that this is also the only used meteorological variable. One reason is simplification, and the other is that it is sometimes difficult to obtain accurate weather predictions. Typically nonlinear combination can be observed between the temperature and the load. As it is stated in [89] that the relationship is highly complex and depends on the geographical region and specific climate characteristics. The usual approach to STLF uses weather forecasts as input features. If multiple weather predictions are available, some researchers report using ensemble approaches in order to use various weather forecasts [90].

- **Static datetime:** Another type of data that is usually not mentioned in literature are static data. In contrast with previously mentioned categories of variables, which are dynamic data (streams), static data are fixed in size and do not receive any new updating records. From the related work we can observe three typical seasonal cycles in load time series data: yearly, weekly and daily seasonal cycles [85]. Most of the yearly fluctuations are the cause of climate conditions, such as outdoor temperature and daylight. In STLF prediction horizons are substantially shorter than the length of the annual cycle, therefore most of the researchers ignore this cycle and focus more on weekly and daily cycles. Due to industrial demands, weekly cycle shows two main groups: week days and weekends. It is shown that weekends also tend to influence the neighbour days. Holidays show similar characteristics as weekends, therefore it is advisable that they are taken into consideration.

In this section, all the features that were identified as possibly relevant are presented, later we evaluate individual features and try to find the most optimal feature set. The entire feature set is presented in Table 3.3. The table consists of the names of data sources, data parameters, unit of measurement for a given parameter, and value of a data stream where *X(t)* presents the value of data stream at time *t*, and *X(t+h)* presents the value of data stream for the time of prediction horizon. Since we know that the measure from yesterday, or previous week, for the same time can be similar to current, and therefore useful for the model, some past measures (1 day, 2 days, and 1 week back) were also included as features: *X(t-1d)*, *X(t-2d)*, *X(t-1w)*.

Since the exploratory analysis showed the importance of calendar trends and patterns, some additional static features were added, such feature is *hour_of_day*. We assume that such feature will be informative for the predictive model since load consumption can be similar for the same hours during working days. That is why *weekday* and *weekend* features were also added. Additionally we have also included holidays, in the form of *business_day* feature.

To incorporate also different short-term trends from the sensor values, we added various autoregressive features, computed for different rolling/sliding window sizes. Relevant aggregate functions are mean *(MA)*, minimum *(MIN)*, maximum *(MAX)*, sum *(SUM)*, variance *(VAR)*. Time windows are labelled with *h* (hour), *6h* (6 hours), *d* (day), *w* (week)

and m (1 month = 30 days). From Table 3.3, we can see that we have created a feature vector with 86 elements. Since the feature dimension highly affects the computation time of certain learning algorithms, we try to evaluate the importance of individual features. We will use this information later to reduce the dimension of the feature vector, by keeping only the most informative features in order to speed up the models' learning and prediction process, while maintaining its performance.

Table 3.3: Entire feature vector schema used in the modeling section (see Section 3.3.5) is presented in this table. We include real-world data from four different data sources: actual readings from different smart meters (AR), current weather (WC), forecast weather (WF) and static datetime features (DT). Features are engineered by calculating different streaming aggregates over sliding windows (such as moving average, minimum, maximum, variance and sum). Additionally historical values, or prediction values (weather prediction), were added. Altogether, 86 features were created which we evaluate in Section 3.3.4.

Source	**Parameters**	**Value**	**MA**	**MIN**	**MAX**	**VAR**	**SUM**
Sensor (AR)	power load	X(t, t-1d, t-2d, t-1w)	h, 6h, d, w, m	d, w	d, w	6h, d, w	-
Weather (WC)	temperature	X(t)	h, 6h, d, w, m	d, w	d, w	h, d	-
	wind speed	X(t)	h, d	-	-	h, d	-
	wind direction	X(t)	-	-	-	-	-
	humidity	X(t)	h, 6h, d, w, m	d, w	d, w	h, d	-
	pressure	X(t)	h, 6h, d, w, m	d, w	d, w	h, d	-
	global radiation	X(t)	h, 6h, d, w, m	-	-	h, d	-
	diffusive radiation	X(t)	-	-	-	-	-
	precipitation accumulation	X(t)	-	-	-	-	-
Weather forecast (WF)	temperature	X(t+h)	-	-	-	-	-
	wind speed	X(t+h)	-	-	-	-	-
	wind direction	X(t+h)	-	-	-	-	-
	humidity	X(t+h)	-	-	-	-	-
	pressure	X(t+h)	-	-	-	-	-
	global radiation	X(t+h)	-	-	-	-	-
	diffusive radiation	X(t+h)	-	-	-	-	-
	precipitation accumulation	X(t+h)	-	-	-	-	-
Static datetime (DT)	hour of day	X(t+h)	-	-	-	-	-
	day of week	X(t+h)	-	-	-	-	-
	month	X(t+h)	-	-	-	-	-
	weekday	X(t+h)	-	-	-	-	-
	weekend	X(t+h)	-	-	-	-	-
	business day	X(t+h)	-	-	-	-	w
	holiday	X(t+h)	-	-	-	-	w
	day before holiday	X(t+h)	-	-	-	-	-
	day after holiday	X(t+h)	-	-	-	-	-

3.3.4 Feature evaluation

As it can be seen from the previous section, feature engineering as a data enrichment process can quickly result in very large feature vectors, which can significantly increase the complexity of the model. Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. The central premise

when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information [113]. Another advantage of reducing feature dimensionality is to enhance generalization and avoid over-fitting (to a certain dataset, or a problem). There are several different known feature selection techniques [114]. One of them is by calculating feature importance scores and using this information to discard irrelevant features. Importance scores can be obtained by using weights from some of the estimators, such as a linear regression algorithm or a random forest regressor. In this analysis we use the random forest regressor[2] to compute the feature importances for all AMI nodes from the grid, for 24 different prediction horizons (from 1 to 24 hours into the future).

Figure 3.9 shows top 20 features with highest importance scores averaged over all available nodes and over all prediction horizons (from 1h to 24h into the future), where 1 is the most important and 0 the least important. Results show that on average, *measurement_1w_back* is the most important feature, which makes sense due to the weekly cycle. The idea behind is that the forecasting value in the future will be similar as it was exactly one week ago. A similar idea holds for the second most important feature, which is *measurement_1d_back*. Overall, we can observe that the most numerous features are the sensor measurements and their aggregates. We can also see four important values from the datetime data source; *business_day*, *hour_of_day*, *holiday*, and *day_of_week*, among the 20 most dominant features. Weather features and their aggregates can also be found in this stack, but interestingly, neither of them has significant importance, since they are not among top 10 most important features.

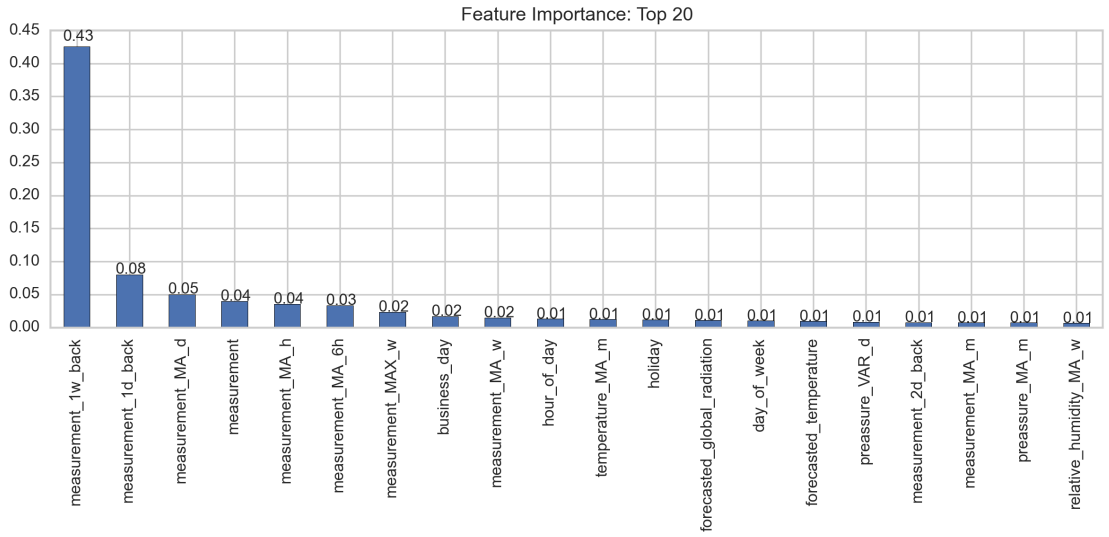


Figure 3.9: List of top 20 most weighted features, where y axis denotes the importance (higher the better). Results are averaged over all available nodes and over all prediction horizons (from 1h to 24h into the future). Result shows that on average, *measurement_1w_back* is the most important feature, which makes sense due to the weekly cycle in the data.

Additionally, heatmap in Figure 3.10 offers an even deeper insight, since it reveals importance for all 24 prediction horizons separately. Horizontal axis presents the prediction horizons, while the vertical axis presents top 20 features (ordered from top to bottom), and

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

every cell contains an importance measure. The most noticeable discovery is that feature *measurement_1w_back* is clearly the most dominant feature for all horizons, except for horizon 1. Clearly, when we are dealing with forecasting shorter than 1 hour into the future, current (last) measurement is the dominant feature. For such short horizons, also moving average with a 1-hour time window is considered as important. This feature also becomes important again for 24-hour prediction horizons, which make sense, due to a daily trend. Beside the measurement from 1 week back, measurement from 1 day back, and daily moving average is also considered as significant for prediction horizons between 2 and 19 hours. For larger horizons than 19 hours, also moving average with a 6-hour time window is considered important. We can see that these results are very useful information if we wanted to develop a more focused forecasting model–meaning not for 24 different prediction horizons, but for a specific prediction horizon, e.g. 1 hour into the future.

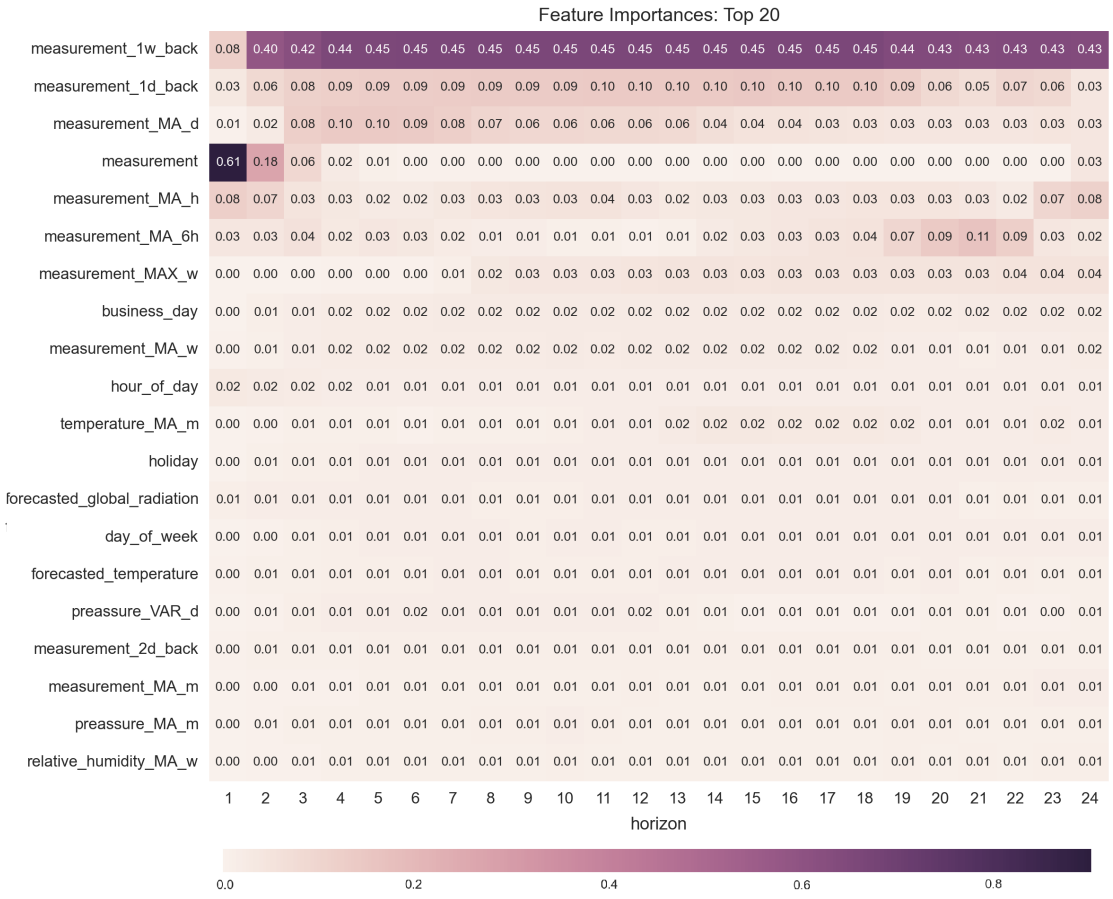


Figure 3.10: Top 20 most weighted features (on vertical axis), separately for all 24 horizons (horizontal axis). We can observe that feature *measurement_1w_back* is the most dominant feature for all horizons, except for the very short-term predictions, where the last measurement (feature *measurement*) is even more important. For mid-term prediction horizons (between 2 and 19 hours) measurement from 1 day back, and daily moving average is also considered as significant. For long-term prediction (larger horizons than 19 hours), additionally moving average with a 6-hour time window is considered important.

3.3.5 Modeling

Beside feature engineering, selecting the most appropriate modelling method that works best for our use case is also a crucial decision. Still, the final success is a combination between the model that we chose, the quality of obtained data, and the extracted features in use (feature set). In order to find the best combination, we have to train and test different methods by using different feature sets and by using different error measurements. We used non-parametric, linear models (e.g. *Ridge Regression*), as well as non-linear models (e.g. *K Nearest Neighbours, Random Forests, Support Vector Regression, Neural Networks*) of different complexities.

We used the Scikit-learn library [107] to train the models, except for the neural network method which is from the PyBrain library [115]. Target values are load measurement values (consumption) for a specific measurement node, for different prediction horizons (1-24 hours into the future). Since we proposed a decentralized method, a different model is developed for each measurement node, as well as prediction horizon. Altogether, 720 models (30 nodes * 24 horizons) were developed for each modelling method:

- **LR** - Linear Ridge Regression
- **KNN** - k Nearest Neighbours (*n_neighbors*=5)
- **RF** - Random Forest Regression (*n_estimators*=100, *min_samples_leaf*=10, *n_jobs*=8)
- **SVR** - Support Vector Regression (*kernel*='rbf', *C*=10, *gamma*=0.001)
- **NN** - (backpropagation method) (*hidden_size*=100, *learningrate*=0.001, *momentum*=0.1, *maxEpochs*=10)

According to data sources, there are the following universal denominations in the text (which are also used in Table 3.3):

- **ALL** – All feature sets (described in Chapter 3.3.3)
- **AR** - Autoregressive variables (measure and its historical and aggregated values)
- **WC** - Current weather
- **WF** - Weather forecasts
- **DT** - Static date time (calendar) properties

With the feature selection process, two additional groups of features are introduced.

- **TOP_20** – 20 most important features (on average; for all measurement nodes, and all 24 prediction horizons) (see Figure 3.9)
- **TOP_10** – 10 most important features (on average; for all measurement nodes, and all 24 prediction horizons) (see Figure 3.9)

For example, the dataset with name *AR_WC_WF* means that autoregressive, current weather and forecast weather features are included.

3.3.5.1 Learning curves

A learning curve is a graphical representation of the models' performance (vertical axis) vs. the experience (horizontal axis), which in machine learning is usually the number of training examples used for learning. Visualizing learning curves offers us a better look at the trained models and allows us to diagnose whether our model is trained well, or if it has

some weaknesses and could be somehow improved (e.g. optimizing method parameters, including more data, reducing the number of features etc.).

Learning curves are presented for several combinations of feature sets and modelling methods in Figure 3.11. We can see that the learning curve of Ridge Regression model is very horizontal, which means that it needs very few training examples to be learned. Based on the accuracy score (R2 score) and feature sets, we can also see that date-time features are very important features for this method (as model accuracy with the feature set that does not contain datetime features (*AR_ WC_ WF*) is substantially lower than with feature sets which contain DT features).

A high gap between training and cross validation test scores usually indicates that the model is dealing with a high variance (over-fitting) problem. We can observe such an example with the *SVR* and *KNN* model in Figure 3.11. We can see that the performance with the training set is perfect, but it does not generalize well with new data (testing set). In such cases, we might improve our model by obtaining more training examples, or by decreasing model complexity (by decreasing the number of features, or model parameter optimisation). We can observe that decreasing the number of features (*TOP_ 10* feature set) indeed helped to reduce the problem of high variance.

Tight fit between training and cross validation scores can indicate that the model suffers from high bias (is under-fitted). This can be seen with *Neural Networks* and *Ridge Regression* model. A consistent gap between the training and testing set indicates a model has learned as much as it can about the data (additional data would not help). One standard way to improve the performance of the model that is suffering from the high bias is by adding additional informative features, or by optimising model parameters. Indeed, we can observe that the accuracy score has slightly increased for both methods when using all features (dataset *ALL*). While parameter optimisation should additionally improve the result at least with the *NN* model.

Ideally, we want to find a sweet spot that minimizes bias and variance, by finding the right level of feature and model complexity. We can observe that the *Random Forests* model seems to be well trained since it does not suffer from the above-described problems. The prediction score is the highest (for both, training and cross validation scores) and it converged to a constant value, with more or less all datasets. A tight gap between the testing and training set also indicates that the model generalizes well with new data.

3.3.5.2 Training times

Models developed in the previous section are of a different complexity, which is highly correlated with computation speed (training time and prediction time). Therefore, model complexity is also one of the important factors that we take into account when selecting the final method to be deployed.

Results in Table 3.4 clearly show that the training time of more complex methods, such as *SVR* and *Neural Network*, is a lot longer than the training time of models that use methods with less complex algorithms. As we would expect, the fastest models are the "simplest" ones, these are *Ridge Regression* model and *k-Nearest Neighbours* models. Training time is extremely important if the model is not supporting online learning (such as *SVR*), and it has to be retrained on all historical data in order to be updated (batch models). Training time also increases even more with higher feature space. We can observe that the training time of the *Random Forests* models clearly increases with increased number of features (as with all other methods), but they are still much smaller than the training times from *Neural Network* and *SVR* models.

For these reasons and reasons described in the learning curve section, we deduce that *SVR* and *Neural Network* methods are not suitable for our use case, where there is a

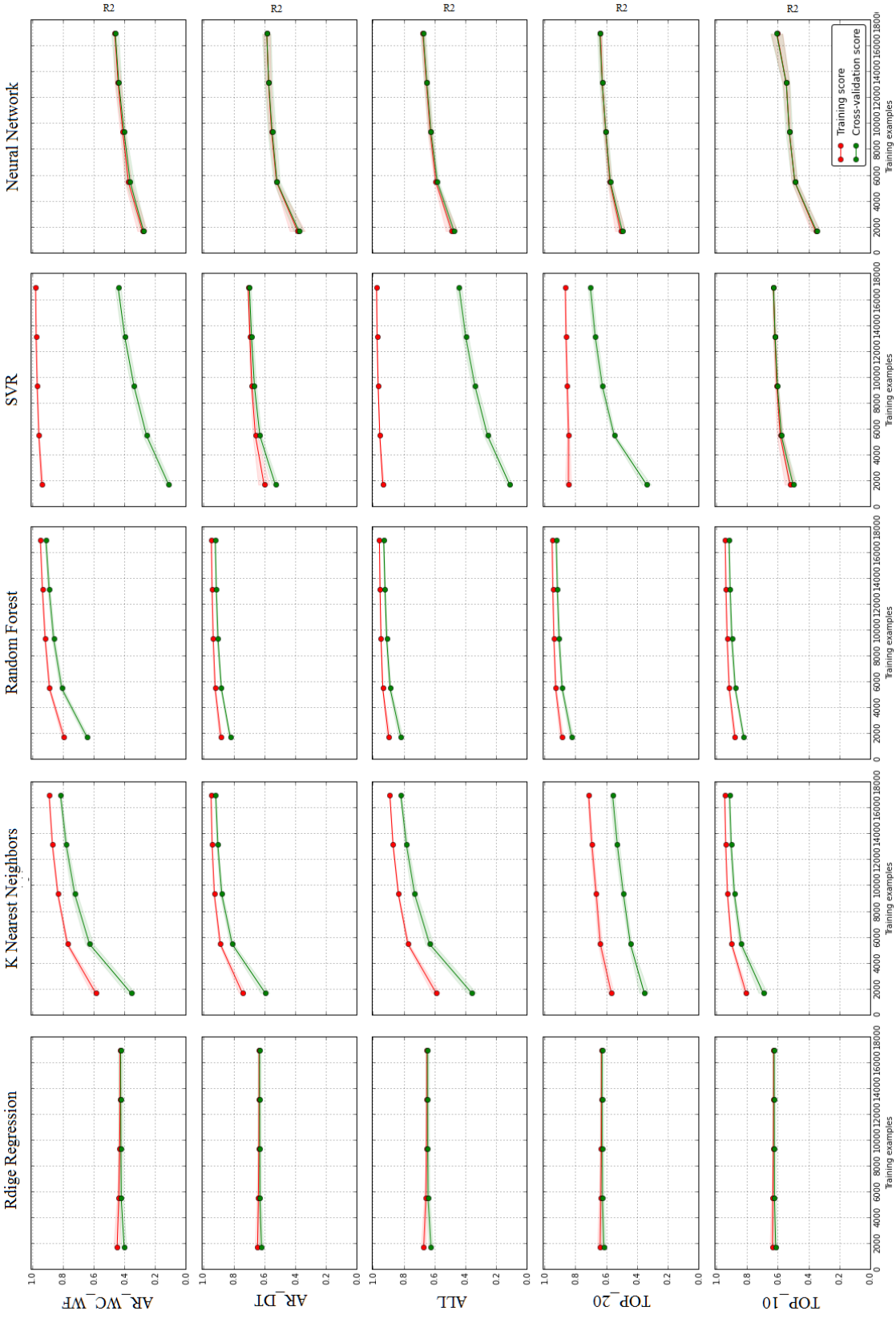


Figure 3.11: STLF models' learning curves for several methods and different feature sets.

need for a fast streaming updating method. In addition, training times of the mentioned methods are also too long (with no increase of accuracy) to be included in the extensive analysis that follows below. Therefore, we proceed with *Ridge Regression*, *KNN* and *Random Forest* models.

Table 3.4: Model training times.

Model	Ridge Regression	KNN	Random Forest	SVM	Neural Network
AR	0.99 s	8.82 s	28.23 s	42 min	23.6 min
AR_DT	1.4 s	27.5 s	31.86 s	1.1 h	24.2 min
AR_WC_WF	1.54 s	9.92 s	132.5 s	6.6 h	25.3 min
AR_WF_DT	1.38 s	25.1 s	54.64 s	5.4 h	33.4 min
ALL	1.96 s	15.2 s	4.8 min	8.9 h	38.4 min
TOP_20	1.57 s	27.5 s	71.19 s	2.4 h	35.1 min
TOP_10	1.59 s	7.54 s	15.54 s	28.6 min	32.9 min

3.3.6 Evaluation and results

Models were evaluated using a 80 % - 20 % hold out testing method approach, where the first 80 % of the original dataset is used only for training, while the last 20 % is reserved for evaluation purposes. In order to put the evaluation results in this section into perspective, we will also compare evaluation results with baseline models and compare the results with state-of-the-art literature. Baseline predictor is usually a feature or a simple naïve predictor that we want to outperform but can still predict our target value well. If the compared method is not better than the baseline, it is usually not much of a use. We use "Last value" baseline, which takes the latest known value from this sensor as a prediction, "Previous day" baseline, which is the value 1 day before the target value, and "Previous week" baseline, which is the value exactly 1 week before the target value.

Figure 3.12 shows the evaluation results, using different combinations of algorithms and feature sets, averaged over all 24h horizons and measurement nodes. Results illustrate that *Ridge Regression* models, developed with different feature sets, do not vary much in accuracy. The two models that differ in performance are the AR and AR_WC_WF. This is due to exclusion of the *"Previous week"* feature, which proved itself to be the most important feature (see Chapter 3.3.4). Further, we can observe that the best baseline provides forecasts with lower (MAPE) errors than the learned models, but the fitness (R2 score) of forecasts to actual value is better with the latter. This indicates the importance of using more than one performance metrics when analyzing the performance of developed models.

The next group of methods was developed with the *KNN* modelling method. According to the results, the best *KNN* models are learned with the low complexity feature vectors (i.e. TOP_20 and TOP_10). These two feature sets contain only selected, most important features. Enlarging the parameter *"n_estimators"* results in higher accuracy with other feature vectors as well, but also significantly slows down the model performance (see Table 3.4).

Random Forest is the third method and according to the results also the one with the most accurate predictions. Results do not vary significantly among the different feature sets (due to inclusion of *"Previous week"* feature—as with *Ridge Regression* models). A bit surprisingly, we can observe that by including current weather features (AR_WC_DT), the accuracy has been decreased. On the contrary, including weather forecasts (AR_WF_DT)

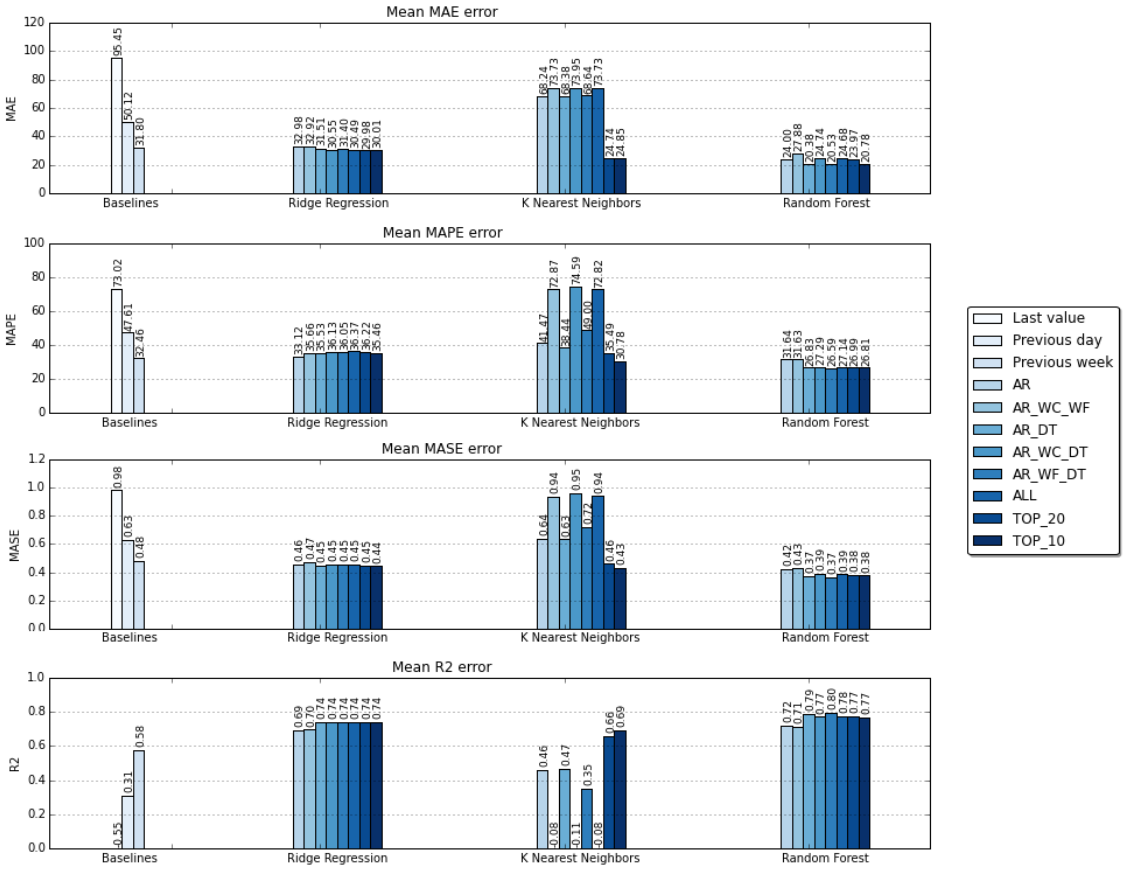


Figure 3.12: Averaged evaluation results (over all 24h horizons and measurement nodes), using different combinations of modeling algorithms and feature sets.

resulted in minor improvement (on average).

We assume that the improvement looks modest due to averaging forecasting results over the large testing period (1 year). When focusing only on shorter periods, with highly variable weather (such as transitions from normal weather to bad weather), the benefit of weather forecasts is more obvious (detailed analysis is presented later in this chapter). Additional error analysis of prediction results shows that the biggest errors occur when predicting from business days (e.g. Friday) to non-business days (e.g. Saturday) and vice versa (see Section A.3 in Appendix A for more details). The main reason is the large statistical differences between these two groups (already mentioned in the data exploratory chapter, Section 3.3.2).

As it was already mentioned that relative forecasting error is highly dependent on the level of aggregation (mean loaf of measurement node) [112], we have evaluated our models also separately for four different nodes with different levels of aggregation (from 1 kW to 0.5 MW). As expected, results (see Figure 3.13) clearly show much higher relative forecasting errors at the individual level nodes (id11010024) than with higher level nodes (id12041022). This confirms the effect of load aggregation on forecasting loads presented in [112]. Therefore, the level of aggregation (or at least mean load) should be taken into consideration when comparing our results to results from other research.

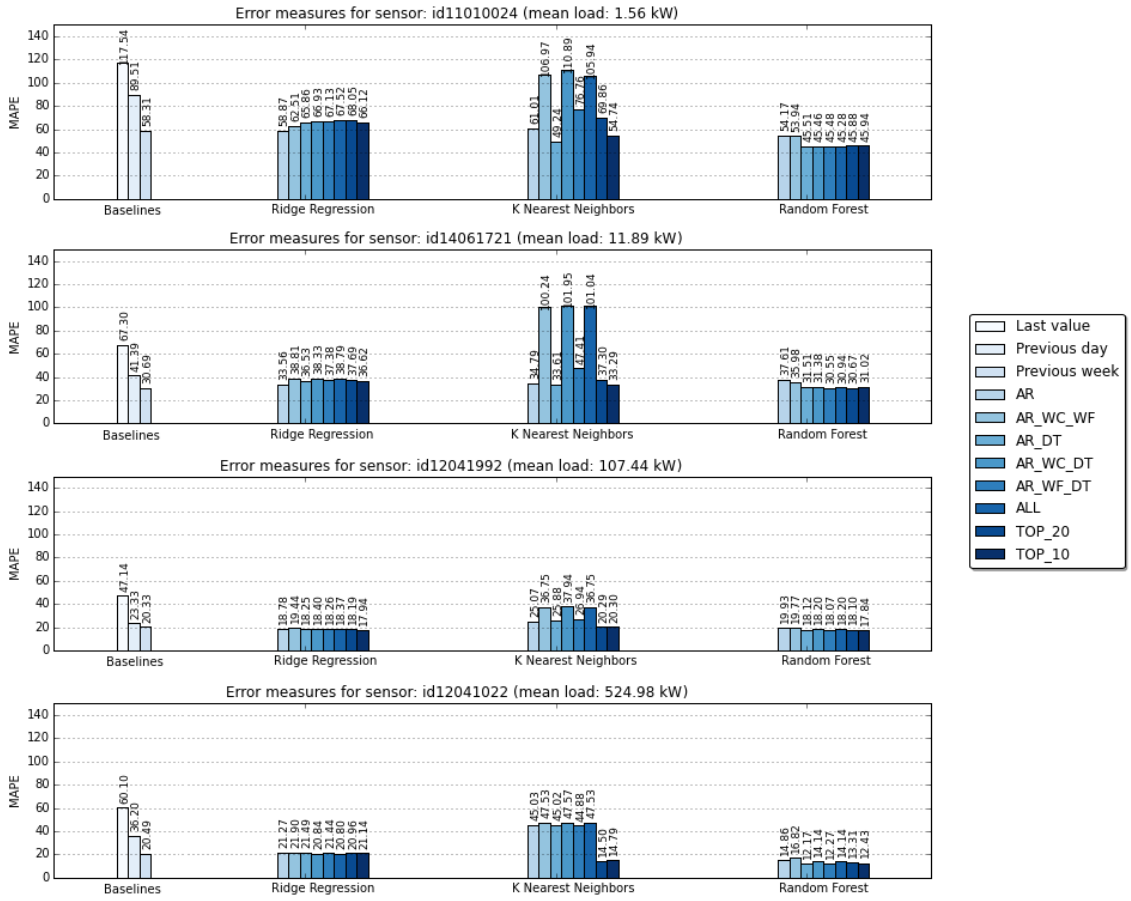


Figure 3.13: Relative forecasting error (MAPE) for each measurement node shows higher errors for lower individual level nodes (id11010024) than with higher aggregation nodes (id12041022).

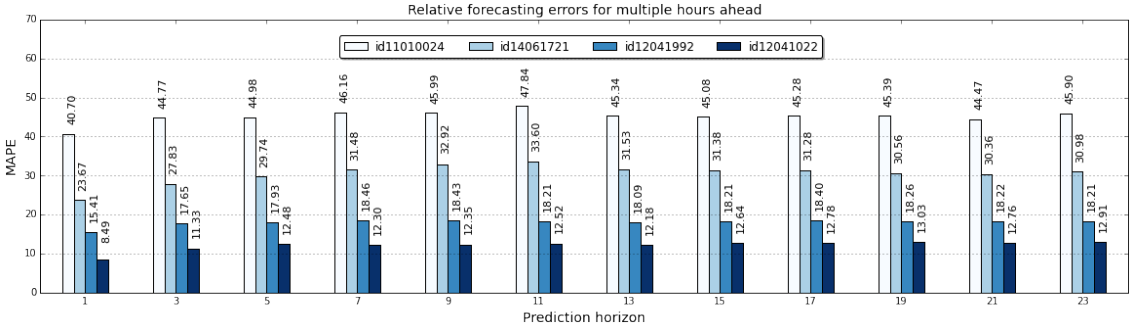


Figure 3.14: Forecasting errors (MAPE) for multiple prediction horizons.

Additionally, we have also separately analysed the models trained for different predictions horizons. Figure 3.14 shows the mean absolute percentage errors MAPE for the Random forest model with $AR\_\ WF\_\ DT$ feature set, for various prediction horizons; from 1h to 24h into the future.

As expected, forecasting error increases with forecasting horizon, but a bit surprisingly, the error is not significantly increased. For example, the error differences between different nodes with different levels of aggregation are much more noticeable which again shows that forecasting error is highly dependent on the level of aggregation (the higher the mean load of the measurement node, the lower the error).

From the results above (Section 3.3.6), it sometimes looks like complex methods—which contained various features from different data sources—hardly outperformed "simple" baseline predictors and even if they did, it was only for some small amount. As it was already briefly mentioned in the previous section, the main reason for such success of baselines lies in the repetitive dynamics (patterns) of the system. Exploratory data analysis revealed repetitive daily, weekly and yearly cycles. Carefully designed baselines can capture this cycle relatively easyly and can estimate the dynamic of the system quite well most of the time when the system is at its "normal" state. The biggest problem of such naïve methods is that they completely fail to predict when the system is not at its "normal" state and behaves a bit differently than usual.

Even though naïve methods' outputs can be very informative (in fact they are widely used in various applications, such as commercial navigational systems due to their computational simplicity), especially for someone who has no previous knowledge about the system (e.g. average consumption at a certain node during the weekday), this information is not interesting for an expert who already has insight into the dynamics of the system at its normal state. In such cases, predictions at "non-normal" situations are much more informative and valuable (e.g. consumption at a certain node during special events). Yet, the evaluation metrics presented in previous sections fail to disclose the models' true predictive value, due to averaging evaluation metrics (such as accuracy) over the entire testing set, and since there are not many "non-normal" situations, the improvement in the evaluation score looks minor, even though it is very significant. In this section, we take a deeper look at the evaluation results and try to assess the models' performance during these so-called "non-normal" situations.

In order to predict the "non-normal" state of the system, the model has to have some "up to date" information about what might have caused it, and has to be able to use it to make a better prediction. When developing our models, we tried to include such information in the form of features from several data sources. In order to demonstrate the significance of our developed models and to stress the importance of additional heterogeneous data sources, we have evaluated models from the previous section also separately over three different

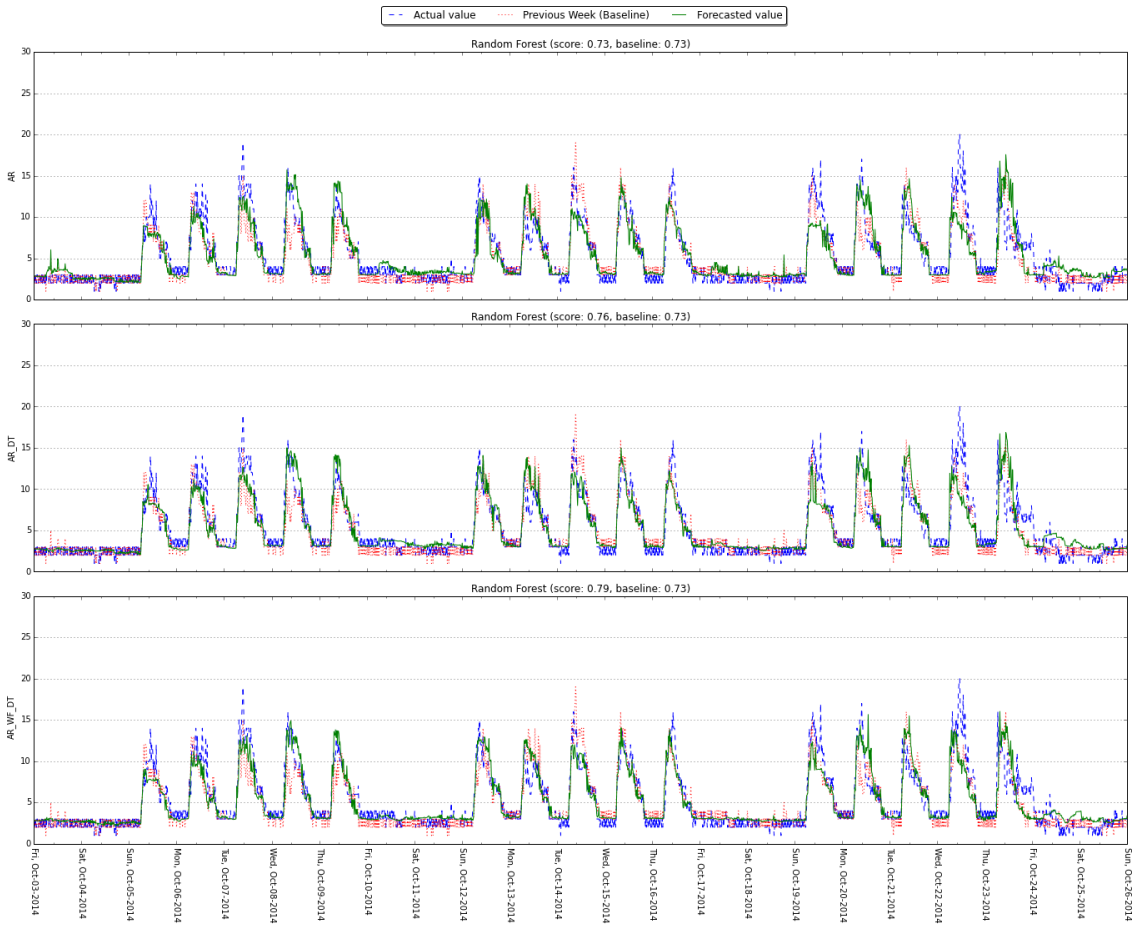


Figure 3.15: "Normal" period evaluation results. Random forest and previous week baseline are achieving similar evaluation results. Additional data sources increase the performance for the Random Forest model.

periods: "normal period", "holiday season", and "bad weather". Random forest models with different feature sets were used to predict 10 hours into the future. Predictions were evaluated by using the coefficient of determination measure (R2) and compared to the "Previous week" baseline.

Figure 3.15 shows the evaluation results over a three-week "normal" period, for a randomly selected node. It is clear how during the "normal" period baseline it is already a very good predictor by itself, and how it is hard to outperform the baseline (by observing R2 metric), even with a more complex method (such as Random Forests in this example). Nevertheless, results show a minor improvement (from 0.73 to 0.79) of the Random Forest model when including additional information such as calendar and weather into the model (feature set AR_DT_WF), which indicates that additional data sources can improve the forecasting prediction. Still, based only on this result, it is unlikely that we would choose a complex method over a simpler method due to its benefits (simplicity, speed, comprehensibility) in such cases (Occam's razor rule).

Evaluation results over a "holiday period" are presented in Figure 3.16. In this case, it is clearly seen how the baseline predictor fails to predict the correct value. R2 result is even negative (i.e. -0.07), which means that the prediction is even worse than simple average over the entire dataset. On the other hand, our random forest model performs much better than the baseline (R2 is -0.07), even by using only the autoregressive feature set which

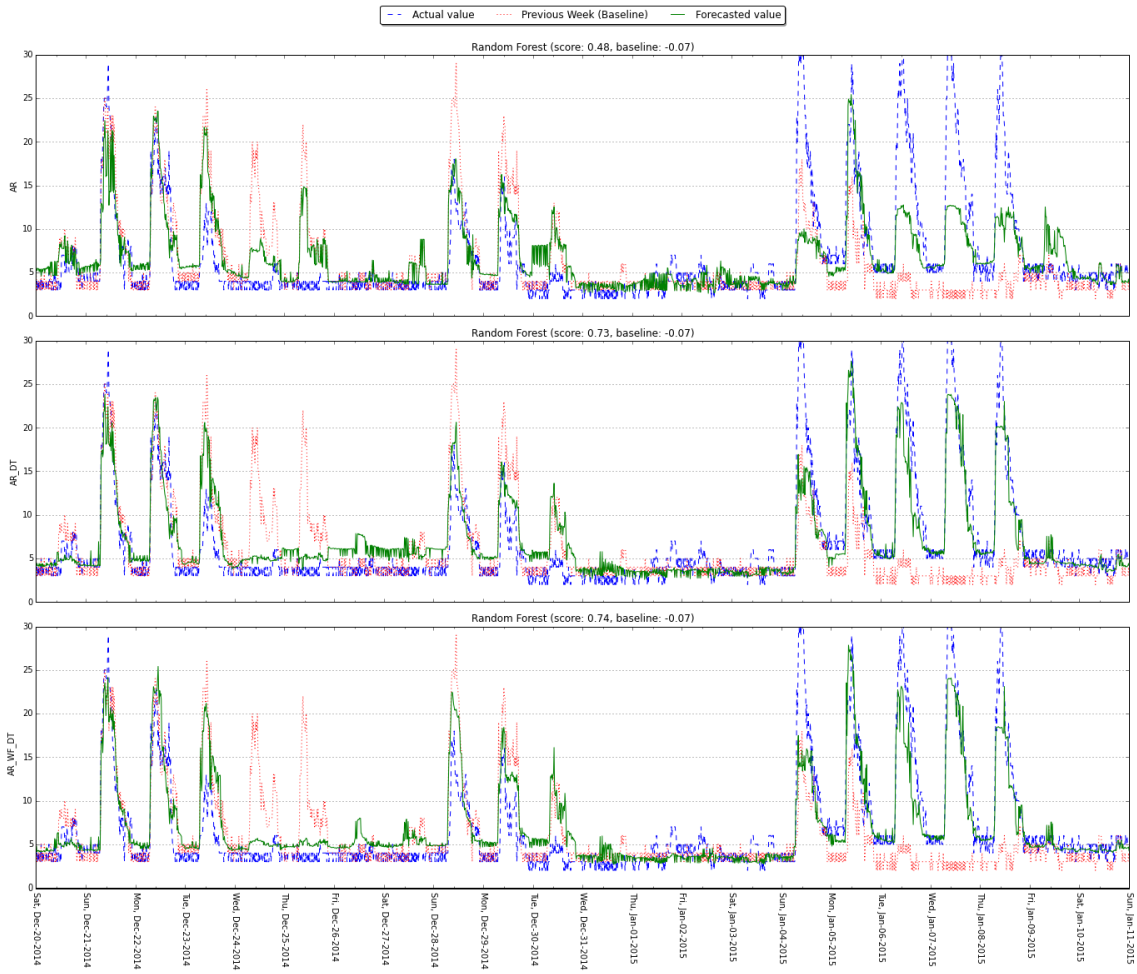


Figure 3.16: "Holiday season" period evaluation results. Baseline fails to predict the correct value, while the model including date time features (AR_DT) shows satisfactory results.

does not contain date time features (with R2 0.48). But, as we would expect, the most significant improvement can be observed when we include date time features (feature set AR_DT) which contain information about the holiday status (R2 from 0.48 to 0.73). By adding additional features from weather forecasts (feature set AR_DT_WF), prediction accuracy slightly increased once more, but not significantly (from 0.73 to 0.74).

The third evaluation period represents a "Bad weather" period (Figure 3.17), during which the average temperature dropped from around 15 °C to around 0 °C (on Monday, October 27th). We can observe the increase of consumption value for this day, probably due to heating. Due to this rapid weather change, we were expecting that additional weather information (weather forecast which also includes temperature) will be beneficial and will improve our model significantly when added to the feature set. A bit surprisingly, the model with the feature set AR_DT_WF scored only slightly better (from R2 0.73 to 0.74) than the same model learned with the feature set without weather features (i.e. AR_DT). This is probably due to the case that such "bad weather" cases happen so rarely that the model could not learn to weight weather features enough, and are therefore not taken into consideration enough when making prediction.

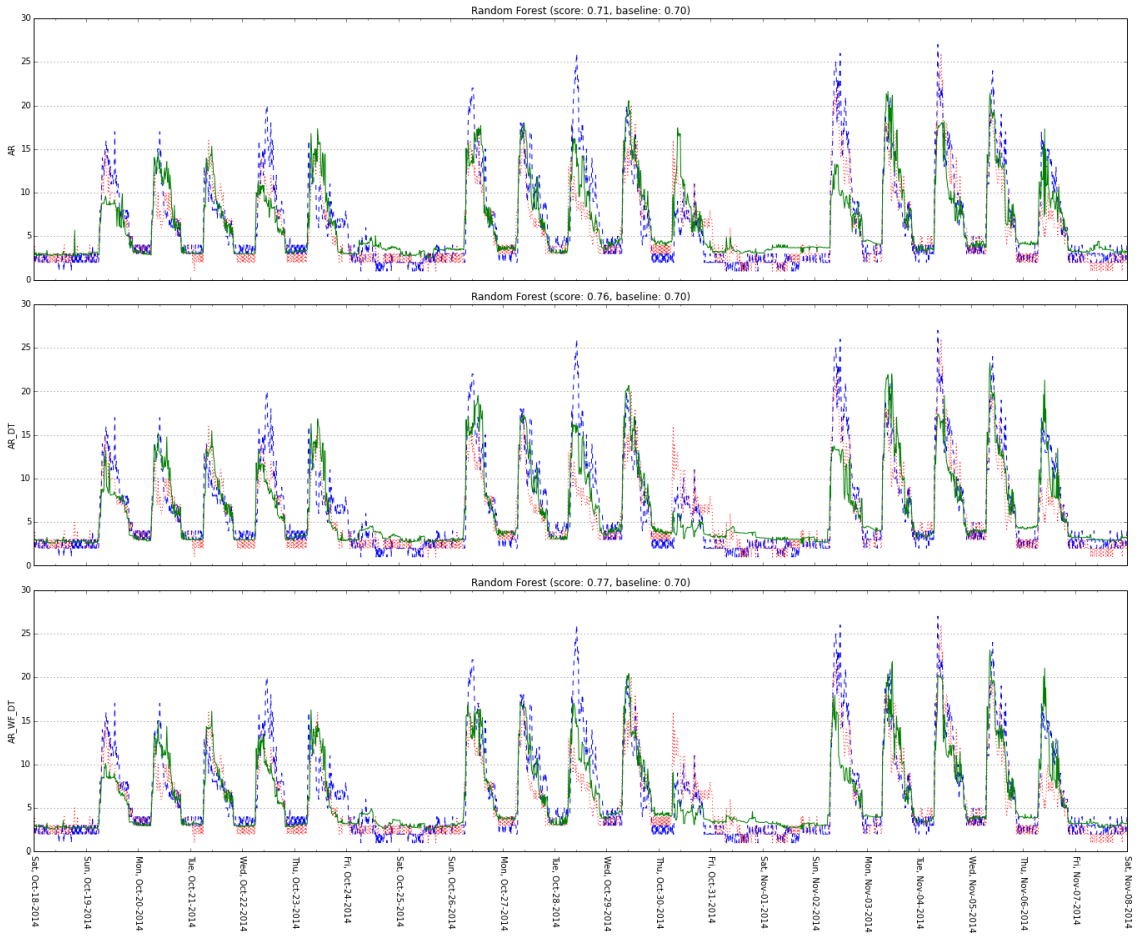


Figure 3.17: "Bad weather" period evaluation results. Only minor improvement in forecast performance for the model including weather features (i.e. AR_DT_WF) than the same model learned without weather features (i.e. AR_DT).

Another explanation is that due to such short-term prediction horizons (10h ahead), weather information is still a bit redundant, since weather changes are not instant (a drastic change will not happen in 10 hours), therefore the weather situation is sort of already included in the current consumption value (current values and very short-term future values are correlated). But, in case of larger prediction horizons, such as 10 days ahead, weather forecast should be more valuable, since consumption can drastically change in the range of 10 days and current consumption will not be correlated with the consumption in 10 days.

We can conclude that even though the evaluation results in the previous section showed only a small difference between baselines and our developed methods, the main advantage of complex methods is the ability to predict in "non-normal" situations which are by far the most informative predictions. This is achieved by including highly informative features into the model from heterogeneous data sources. But the main disadvantage of non-parametric data-driven methods remains, which is the fact that the model can only predict in anomalous situations, which has already been seen in the learning dataset (historical data). Unlike the parametric model, data-driven models cannot make accurate predictions for unseen anomalies.

3.3.7 Offline, batch and online models' performance

In this section we present the results of comparing the performance of models using the three methodologically different approaches; *offline*, *batch* and *online* models (described in the introduction of Chapter 3). For the comparison between the approaches, we used the algorithms from Scikit learn (described in Chapter 3.3.5) for the offline and batch approach. We simulated the batch approach by using the classic offline approach, but with adding the additional new datapoint to the existing dataset and we retrained the model, therefore this model has seen all the data from the past, similar as the online version which only has to be updated and not retrained (see Figure 3.18). Since Scikit-learn library does not provide the online versions of the selected algorithms (Linear regression, KNN and Random Forests), we had to use other implementations. For online version of linear regression, we used the implementation from QMiner library, i.e. incremental recursive linear regression[3]. Data drive KNN version was implemented by using the MOA library [48]. For online version of Random Forests, FIMT-DD (Fast Incremental Model Trees for Drift Detection [64]) algorithm from MOA was used[4]. Since we use different implementations of offline and streaming algorithms, the following comparison is a bit unfair, but still some general conclusions can be made.

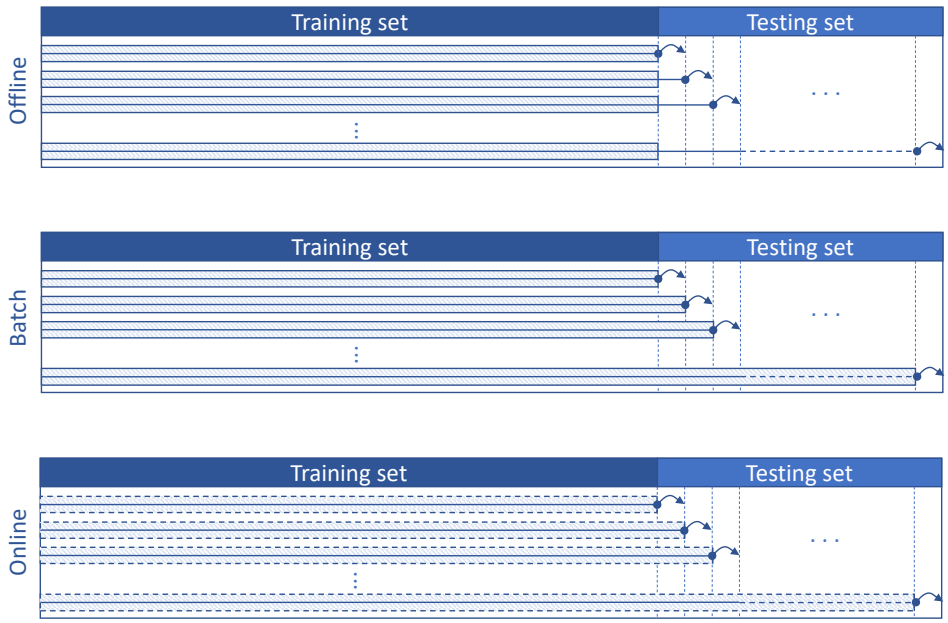


Figure 3.18: Offline model was trained by using only the testing set. In the batch model we incrementally add additional new datapoints from the testing and retrain the model before prediction. With the online model, retraining is not necessary as we can just update the model with the new datapoint.

Figure 3.19 shows the comparison results between the methods by using the R2 evaluation metric, averaged over all 24 prediction horizons, for the same four sensors as in previous chapters. Results show very similar evaluation results between the three approaches. In most cases, the batch model performed the best, since it was trained with more training data than offline, it consequently included also the latest data, which are the most im-

[3] `https://rawgit.com/qminer/qminer/master/nodedoc/module-analytics.RecLinReg.html`

[4] `https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/classifiers/trees/FIMTDD.java`

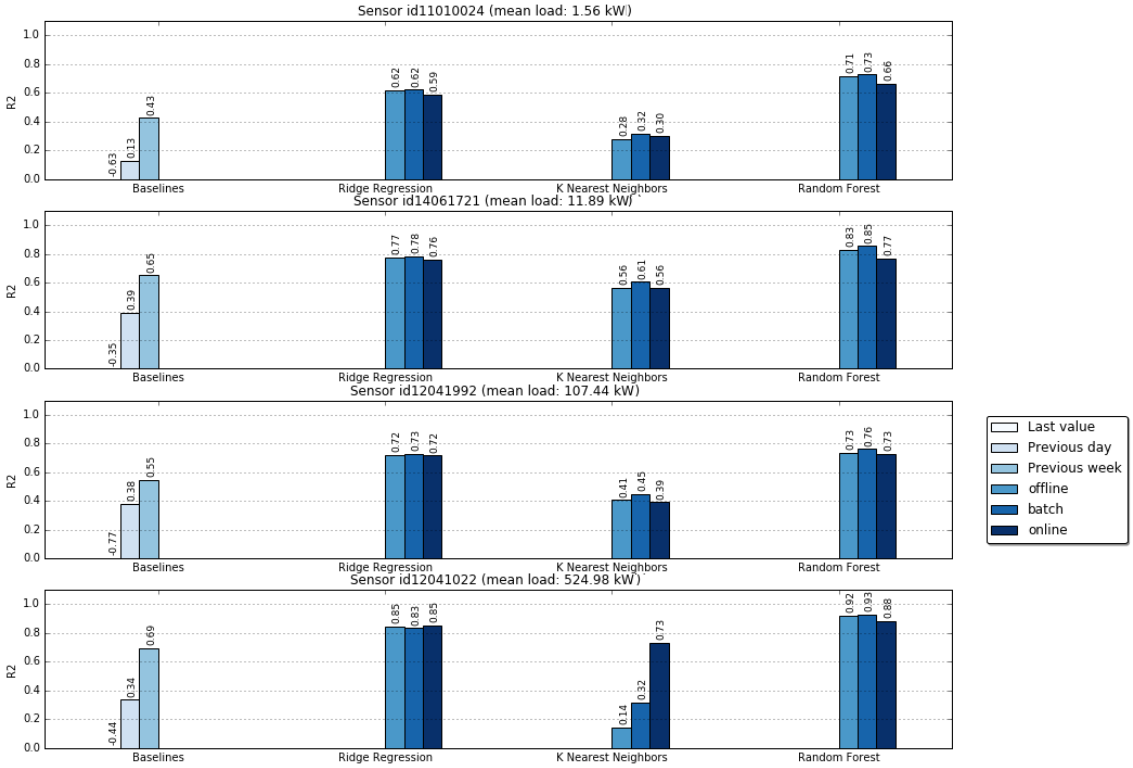


Figure 3.19: Comparison of offline, batch and online models, using averaged R2 evaluation score over all 24 prediction horizons, applying AR_WF_DT feature set with different modeling algorithms.

portant information for successfully handling concept drift situations. Additionally, the batch approach has the advantage over the online method to re-learn the model at each new data using the entire dataset (and not just update the model), due to which it can calculate better parameters and can construct a more optimal model structure. It can also weight recent data points more, so that they have a larger impact than older data, which is especially good for concept drift situations. However, the latest trends in the field show the new algorithms which can also detect the concept drift (such as FIMT-DD [64] and iSOUP-Tree [65]).

Additionally, Figure 3.20 demonstrates how prediction performance changes over time as it gets more data and compares all three approaches, using the random forest models. Results represent the evaluation scores over the testing dataset, with the regression plot trend calculated for each approach. As expected, the batch approach performs better than the offline approach, since the model was re-trained at each new data. We can observe that the performance trend for the online approach is also slightly better than the offline approach (since it was updated with each new record), but still worse than the batch approach. Taking into consideration that the online versions of regression model trees with drift detection are still very new and still have some room for improvement, the results make sense. Nevertheless, the results show that the performance of the purely incrementally learned models is comparable to classical batch learned models, which is in practical terms a very positive result due to other advantages of online streaming algorithms, such as speed and computational efficiency). While the the computational time increases linearly with the number of records for offline and batch approach, the online approach latency stays the same (see Figure 3.21), meaning that the learning efficiency is much better.

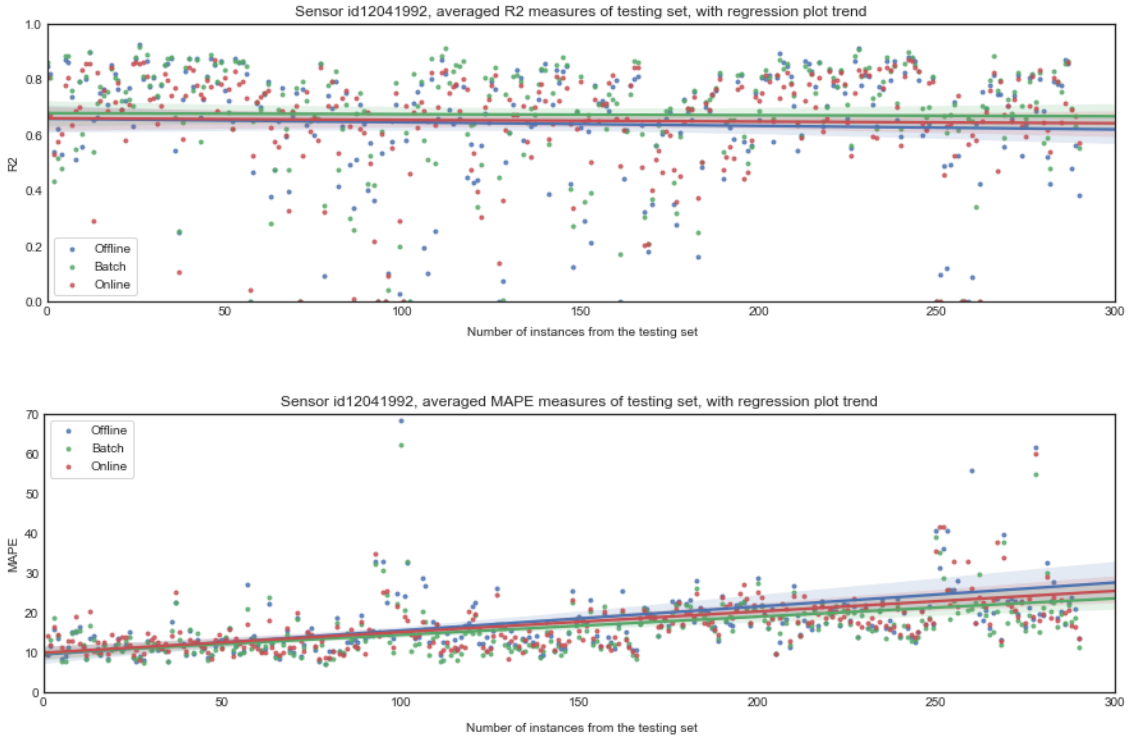


Figure 3.20: Comparing model trees performance over time for offline, batch and online approach (averaged over all 24 prediction horizons for sensor id12041992).

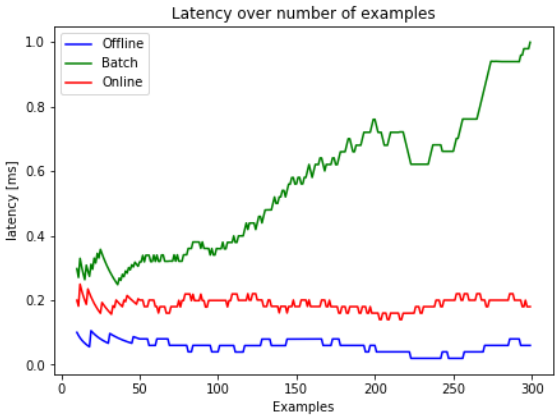


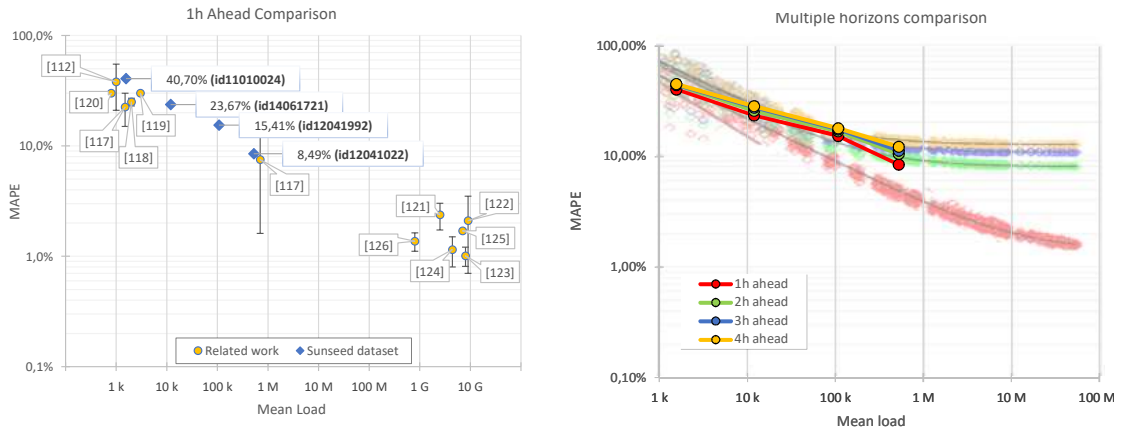
Figure 3.21: Comparing different modeling approaches and prediction efficiency. The test was done only for demonstration purposes, for one prediction horizon, using the linear Stochastic Gradient Descend regressor (which is the only regression method in Scikit library with the option of incremental learning).

3.3.8 Comparing models' performance with the literature

In this section we compare our results with some of the state-of-the-art research from the literature. We should point out that this is not really a fair comparison, as many factors between different research differ. To start with, the datasets used for learning and evaluating the models are different between the compared research. As we know, quality as well as quantity of data greatly affects on the models' performance. Evaluation metrics, level of aggregation, prediction horizon window, data pre-processing steps are all additional factors that differ among research used for comparison and affect the final evaluation [116].

For the sake of comparison, we include the literature short-term load forecasts with 1 hour-ahead prediction horizons (see Figure 3.22a). The models with mean load around 2 kW achieve MAPE score which ranges all between 25 % and up to 50 % [112], [117]–[120]. Our results, for a comparable node with a mean load of 1.5 kW also falls into this group with 40 % MAPE, which is comparable to the reported literature. The next cluster is the literature, where the main focus are nodes with the highest level of aggregation (i.e. transformer or generation stations with thousands of users), where MAPE is substantially lower due to high aggregation—between 1 % and 3 % [121]–[125]. The closest result to our highest node (i.e. id11010024 with a mean load of 525 kW), was explored in research [126] (commercial building with a mean load of 700 kW), where the authors reported MAPE between 1.61 % to 13.41 %, which is also comparable to our result of 8,49 %.

Even though our results are more or less comparable with the literature, one of the main reasons why they are not better is the fact that in Figure 3.22a we were comparing prediction results for 1 hour ahead, while our model (i.e. selection of feature set) was designed to be generic enough for prediction horizons between 1h and 24h, meaning that we included all the features that proved to be useful for 1h prediction horizons as well as 24h prediction horizons. Optimizing the feature set only for predicting 1h ahead would probably increase the prediction accuracy for this prediction horizon. Figure 3.22b shows the comparison of our results with [112], where the authors present results for various aggregation levels (from 1 kW to 100 MW), for four different prediction horizons, i.e. 1h -



(a) (b)

Figure 3.22: Comparing our results with the state-of-the-art literature. (a) Comparison of load forecasting MAPE scores for 1 hour into the future. (b) Shows the comparison of forecast evaluations with results from [112], for multiple prediction horizons, that is 1 - 4 hours ahead. Note that for the sake of comparison our results are presented over the figure originally published in research from Sevlian et al. [112].

4h ahead. We can observe that our results for prediction horizon 2h and higher are even more in line with the literature than 1h ahead predictions.

3.4 Application

One of the major goals of the SUNSEED project[5] [110] was to improve the observability of the distribution grid, which is one of the key challenges in the context of introducing smart grids. From the distribution system operators' perspective, observability of the electricity distribution grid is particularly important to allow higher penetration of distributed resources into the grid. To avoid high investment costs into the reinforced grid, the grid requires some smartness, where all decisions are based on the in-real-time estimated state of the grid, as well as on the forecast state of the grid. By closely following the operation of the grid, potential disturbances can be identified and eliminated at the place of their origin. For this reason, a three-phase state estimation model was developed during the project [111]. Furthermore, a forecasting module enables additional possibilities for control and future planning, offering applications, such as load switching and optimization, decision support for power systems operations, maintenance planning and trend detection.

Forecasting module is one of the crucial building blocks in all smart grids. We have developed a decentralized, data-driven, streaming forecasting model capable of processing a large amount of real-time heterogeneous data streams and producing predictions on various nodes in the grid, from 5s to 24h into the future. In order to build the most effective model for the specific power grid (they differ in geographical properties, meteorological factors, grid topology, etc.), key outcomes of the exploratory data analysis were taken into consideration.

Architecture

The final streaming prototype was built on top of the QMiner[6] open-source framework [49] and supports processing large-scale real-time data streams from multiple sources in an online fashion. The prototype consists of three main streaming components (see Section 3.2.2): (i) Stream processing (responsible for online handling data streams – cleaning, pre-processing), (ii) Fusion (includes a mechanism for online merging and feature engineering), and (iii) Modeling component (includes online regression forecasting algorithms) [127].

The main two data sources for the prototype are smart meter (AMI) measurements which we obtain once per day (on 15 min resolution) and WAMS (Wide Area Measurement System) measurements with a much higher streaming frequency (50Hz – near real time). Beside the real-time grid measurements from the testbed (from AMI smart meters and WAMS meters), two additional external data sources were taken into account: static date-time data (time of day, working day, holiday status, etc.) and weather data (current measurements and forecasts). For each data source, additional streaming data aggregates were developed for the purpose of data cleaning, pre-processing and data enrichment (feature engineering) (see Figure 3.23).

To ensure the scalability of the system, the developed prototype due to its decentralized architecture enables parallelization (running different sets of models on different instances in parallel; in the current setup we are running one instance for 5 seconds event horizon and another for all the other models). Instances can be run anywhere within the closed network where MQTT data is available.

[5] `http://sunseed-fp7.eu/`
[6] `https://qminer.github.io/`

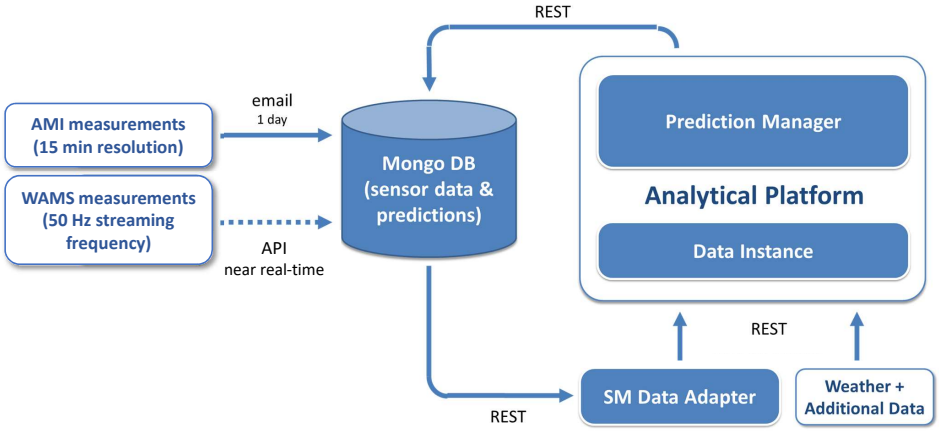


Figure 3.23: The application uses two main data sources; AMI measurements (every 15min) and WAMS measurements (50 times per second). Data is stored into a MongoDB from where the analytical platform retrieves the data and enriches it with additional external data sources, such as weather and calendar. Predictions for various prediction horizons are pushed back to the MongoDb, from where they are used by other Sunseed components and for visualization purposes.

Use cases

The final forecasting prototype is a web application (see Figure 3.24) which contains several decentralized models (for each sensor node from the low- and mid-voltage grid), and supports three important project use cases:

- **Short-Term Load Forecasting (STLF)**

  The main data source for the STLF are smart meter (AMI) measurements, which are primarily used for billing purposes and therefore obtained only once per day via email, for 750 different nodes. Based on this data source, the model is able to forecast the load for a specific node for various prediction horizons: from 1h to 24h into the future. Since computation speed is not of key importance in this use case, Random Forest model was used due to its performance superiority. These predictions can be used by the distribution system operators, as decision support for power systems operations and as up-to-date estimations of the grid for the state estimation module.

- **Very Short-Term Load Forecasting (VSTLF)**

  The main data source are WAMS (Wide Area Measurement System) measurement units, with a near real-time streaming frequency of 50Hz, which generate vast amounts of data (1GB per day per unit). In this case, the model is calculating very short-term predictions – from 5s to 15min into the future, for 16 different nodes. Simple and effective recursive linear regression was used in this use case, due to its computational effectiveness (i.e. speed) and satisfactory evaluation results. Such extreme short-term predictions are useful for autonomous load management applications and various control system modules.

- **Exploratory analytics**

  Use case provides several exploratory analytics views useful to grid operators for grid management, planning and maintenance. The user can visualize historical data over different time windows, and can compare them with various aggregates (e.g. moving

averages, local minimums and maximums over certain time windows). Weekly and daily consumption profiles (histograms) are calculated for all nodes once per day, which gives an expert user additional introspection into the behavior of a certain consumer. Additionally, application also provides temporal visualisation of the availability of obtained data from sensor nodes, with geographical visualisation of location on the map.

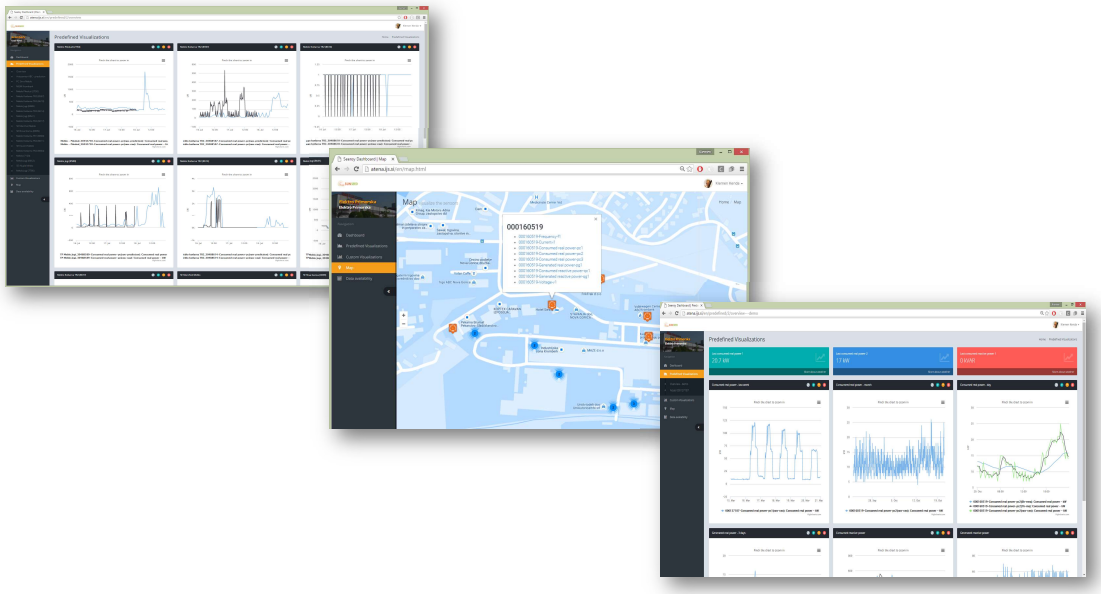


Figure 3.24: Sunseed application covered various use cases: short-term load forecasting (with predictions from 1h – 24h), very short-term load forecasting (with predictions from 5s to 15min), and exploratory analytics views useful for grid management, planning and maintenance.

Chapter 4

Conclusions

This work propels the field of smart analytics for smart cities by developing a novel method for modeling human mobility, which goes beyond predicting only the users' next location. To the best of our knowledge, this is the first published method that enables predicting multiple sequential locations into the future, including the time of arrival and residence time – i.e. predicting users' entire mobility patterns. We also present a generic framework for fusioning and processing streaming sensor data obtained from various data sources. By using the proposed approach, we successfully developed and evaluated a model for electricity demand forecasting by using real-world data.

4.1 Contributions to Science

A novel light-weight approach for predicting human mobility patterns (next location prediction) was presented in this thesis (C1). Our proposed approach (TMAP, see Section 2.3.2) is based on a statistical Markov state-space model (frequently used in this research area), enriched with additional temporal features; i.e., *arrival profiles* and *probability of stay*, which enables dynamic predictions of the next location and time of arrival (prediction changes over time) and reduces the bias of the basic Markov state model of predicting only the most frequently visited locations. This enhancement also increased the prediction power (i.e., *recall*) of the proposed method, as the model also enables the prediction of the less frequently visited locations, based on the information of time of interest. By using this additional temporal information, the model also tends to produce more meaningful and informative predictions. For example, the model can quickly learn the commuting habits and takes into account that it is very unlikely that a person will visit a location, such as *"work"* in the evenings (unless this is the actual users' habit), even though *"work"* is one of the most frequently visited locations. Another advantage of this method is that it is computationally less demanding. Consequently, we can calculate the predictions for each individual "on the fly" at the time of interest. This also makes the model very scalable (using distributed computations) or it can also be used directly on the end users' devices (e.g., smartphones).

We extend our proposed method (MC model, see Section 2.3.3) with the use of Monte Carlo simulations, which enables us to predict users' longer-term mobility patterns (C2). By being able to model humans' mobility patterns, this also opens up the possibility of answering more complex mobility-related questions, where we have to take into account that the user can change more than one location before arriving at the final location at a particular time. These are questions such as *"What is the most probable users' location at time hh:mm?"*, or *"What is the probability of being at location X at time hh:mm?"*. Naturally, we can assume that answering such questions would be beneficial in many business

cases, such as direct target marketing, where it is very informative if we know what is the most likely time for the person to go to the store, fitness, or to use public transit, and similar examples.

We extensively evaluated the proposed methods on real-world scenario use cases, using real-world datasets (C3). Regarding the proposed next location prediction model, we conclude that by focusing solely on prediction accuracy, no significant improvement can be reported when comparing results to the related state-of-the-art research. However, additional evaluation measures (recall and F1) highlight the increase of prediction power of our proposed model (higher recall). Evaluation of the external dataset with hundreds of users reveals that the performance of our proposed model is highly correlated with the quality of available data for specific users. Good quality data allows us to construct informative temporal features (arrival profile and the probability of stay), which improves the prediction.

The results also showed that the higher-order models O(*2*) do not improve the models' performance enough (in comparison to O(*1*) models) to justify the significant increase of computational complexity. Regarding the short-term load forecasting use case, we model the energy consumption dynamics of smart grid consumers by using autoregressive feature vectors enhanced with information from outside data sources, which enables the use of data-driven machine learning methods to predict the future. The models' performances were in line with the most current and relevant literature. Comparing the results of offline, batch, and purely online approaches shows that the performance of purely incrementally learned models is comparable to batch-learned models (where computational time increases with the amount of data). Overall, the results prove that the proposed approach of designing streaming models is beneficial and sufficiently robust for industrial applications (also presented in the thesis).

4.2 Hypothesis Assessment

By evaluating our proposed models, we have confirmed our first hypothesis (H1) that including multiple relevant data-sources indeed improves the performance of the model. Even though this might not be very clear at the beginning (since the improvements can be quite minor over the larger period as the dynamics can be highly predictable most of the time), we observed that the biggest improvements were observed during anomalous events (such as severe weather change), when the accurate forecasts are actually the most beneficial. Effective feature extraction clearly has a great impact on model performance when predicting anomalous situations, as well as the choice of the learning algorithm. Even though light-weight modeling algorithms can be very efficient most of the time, in general, more complex algorithms (such as Random Forest) tend to detect correlations between data sources better in rare and non-normal situations and are consequently better at predicting future anomalous situations.

Our analysis shows that our second hypothesis (H2), which states the online streaming models should produce comparable results as offline models in real-world scenarios, can be partially confirmed. Although the results show that the predicting performance of online models is slightly worse than with offline and batch models, we still observe a comparable prediction performance. However, the learning efficiency of online models is much better when working with large amounts of heterogeneous streaming data. Therefore, we can conclude that the choice of the modeling approach depends on the use case. In the case of large volumes of fast-streaming data, the online approach would be the best choice (or even offline if we know that statistical properties of the data will remain the same – i.e. no concept drift). However, in case of large amounts of data that are delivered on a much

slower pace (e.g., once per day), the batch approach seems to be the best choice (in model performance terms).

4.3 Future Work

Based on our experimental evaluation, we can conclude that the proposed approaches are effective and useful in real-world scenarios. The outcomes of this thesis are continuously used in various applications used for smart grid management and intelligent traffic applications. However, there is still a lot of room for improvements. As future work, we plan to extend our research in different directions.

One of the main challenges that we encountered in the data (in both use cases) is some sort of a significant change in the data at some point, also known as concept drift problem (a very common occurrence in the real-world data), or non-stationary low-quality data. One possible approach to handle such challenges is to introduce an ageing mechanism into our model (as proposed by [10], [11]), so that the model can adapt to newer data sooner. During the research, we observed that we are in the middle of a highly active period of development of online streaming methods for regression problems with an emphasis on detecting concept drift. One of such state-of-the-art approaches is proposed by Osojnik et al. (iSOUP-Tree [65]). We believe that this new implementation could further improve the performance of the data-driven models in the real-world use cases. Therefore, we plan to evaluate and include some of these new implementations into our modeling pipeline.

Additionally, we plan to include additional heterogeneous data sources into our pipeline, such as extracted relevant information from the news about the relevant current and future events (using EventRegistry [128]). We expect that such information would serve as additional useful information to the models and could further improve the performance.

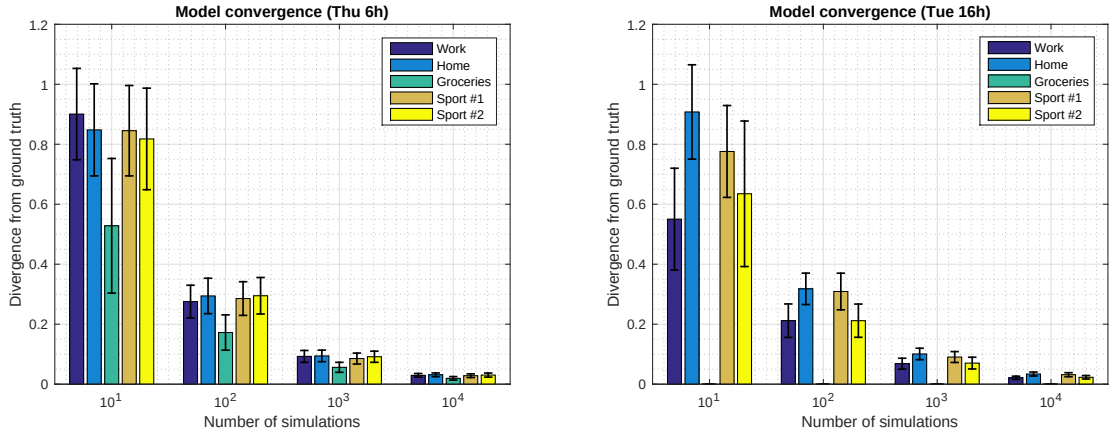
Finally, since we have presented the new method based on Monte Carlo simulation, which enables predicting multiple sequential locations in the future (with the most probable time of visiting the next location and also residence time), it would be highly beneficial to evaluate also the prediction accuracy of more than only the first next location, but also the second location, third location, and so on.

Appendix A

A.1 Convergence Analysis of Monte Carlo Integration Method

To test the convergence properties of our proposed Monte Carlo predictor (described in Section 2.3.3) we measure how various next state probability distributions—obtained by using a different number of simulations (10^1, 10^2, 10^3, 10^4)—differ from the "ground truth" probability distributions (estimated by using 10^6 simulations). For each distinct group (characterized by the same number of simulations), we generate 100 different distributions with the same initial parameters (i.e. starting location and starting date). To measure the distance between these simulated probability distributions and the "ground truth", we use L_1 distance measure, which is simply twice the total variation distance between the distributions.

Figure A.1 shows the average distance and standard deviation over all 100 simulations for each distinct group. Analyses were performed for various starting locations, as well as for different starting dates. We used a sum of absolute errors to measure the divergence between distributions. For each distinct group (with the same number of simulations), we generate 100 different distributions with the same initial parameters, which enabled us to estimate also the standard deviation of the errors, which offered us additional insight into the convergence properties of the predictor. Further on, analyses were performed for different initial parameters; different starting locations, as well as for different starting dates. As expected, results show that the distance from "ground truth", as well as the standard deviation, decreases with the number of simulations. By taking into consideration also the computational time component (which increases linearly with the number of simulations), we decided to use 10^3 simulations per prediction, which in our case seems to be the optimal trade-off between performance and computation time.



(a) MC predictor convergence analysis with initial starting date Tuesday 6 am

(b) MC predictor convergence analysis with initial starting date Thursday 4 pm

Figure A.1: Convergence properties analysis for different initial parameters (various starting locations and starting dates) reveals the expected decrease of distance between the simulated next location probability distribution and the "ground truth" distribution as a function of the number of simulations. Taking into consideration also computation time, 10^3 simulations per prediction were used as a good trade-off between accuracy and computation time.

A.2 Evaluation Metrics

Evaluation metrics are used to evaluate and compare forecasting models for purposes of selecting the most appropriate model for our needs. When dealing with labeled data (supervised learning), we categorise problems into two groups: classification problems and regression problems. These two groups are fundamentally dealing with different problems: classification is predicting a label, and regression is predicting quantity. Therefore different modeling algorithms are being used for each group and also different evaluation metrics are being used for measuring the performance of the models from each group.

A.2.1 Evaluation metrics for classification models

There are many evaluation metrics that can be used for evaluating the classification models. Majority of the metrics are designed to be used on binary classifiers, where there are only two mutually exclusive outcomes possible (see Section A.2.1.1). For multi-class classifiers, where there are three or more outcomes possible, several strategies can be used to reduce multi-class problem into binary problem, so that binary classification metrics can be used (see Section A.2.1.2).

A.2.1.1 Binary classification evaluation

Most of the binary classification metrics are calculated by using counts of the following occurrences: number of *false positives* (FP), *false negatives* (FN), *true positives (TP)*, and *true negatives (TN)*. These values are usually represented in confusion matrix (see Table A.1), which is a special type contingency table with only two dimension: "True condition" and "Predicted condition". Four most commonly used metrics (that we also use in this dissertation) when evaluating classification models are *accuracy*, *recall*, *precision* and *F1 score*. Metrics are presented in detail in Table A.2.

A.2.1.2 Multi-class classification evaluation

One of the techniques to reduce multi-class problems into multiple binary problems, where we can use binary classification metrics is so called One vs All. In this technique we evaluate each class separately, by considering each class as positive and all other as negative. We end up with a separate result for each class. In order to extract a single number from all class results which will determine a models' performance, it is common to use two different kind of averages:

- **macro averaging**: is calculated by simple mean of the binary metrics. Usually used when we have unbalanced dataset and we want to give equal weight to each class when calculating overall metric.
- **micro averaging**: is calculated by summing the dividends and divisors that make up the per-class metrics to calculate an overall quotient. Micro averaging is preferred when we want to bias our metric to majority to the most populated classes.

Table A.1: Confusion matrix table for binary classification.

	True Positive	**True Negative**	Total
Predicted Positive	TP	FP	$TP + FP$
Predicted Negative	FN	TN	$FN + TN$
Total	$TP + FN$	$FP + TN$	N

Table A.2: Metrics for evaluating classification models.

Name	**Formula**	**Description**
Accuracy	$\frac{TP + TN}{P + N}$	Correctly predicted true results among the total number of all cases. Tells us how well the model is performing.
Precision	$\frac{TP}{TP + FP}$	Correctly predicted true positive results, among the number of all positive samples. Used when we want to be very sure about the prediction.
Recall	$\frac{TP}{TP + FN}$	Proportion of actual positives is correctly classified. Recall is used when we want to predict as many positive outcomes as possible.
F1	$\frac{2TP}{2TP + FP + FN}$	Harmonic mean between recall and precision. It is usually used when we want to have a model that is good in precision as well as in recall.

A.2.2 Evaluation metrics for regression models

According to Hyndman et al. [116], four types of evaluation metrics exist for regression modeling:

- **scale-dependent metrics** (RMSE, MAE, ME), good for comparison between the models of the same series, but should never be used to compare models with different series (e.g. data from different sensors), since it makes no sense to compare the accuracy on different scales.
- **percentage-error metrics** (MAPE) have the advantage of being scale-independent and are frequently used to compare the performance between different data series, but still suffer from a problem if there are zero values in a series. This causes division by zero.
- **relative-error metrics** (MdRAE) uses the naïve method as a benchmark and are also scale-independent, but they suffer from the same problem as MAPE – division by zero – when dealing with infinite or undefined data.
- **scale-free metrics** (MASE) can be used to compare forecast methods on a single series and also to compare between series. They are not affected by zero values in the data. The author believes that MASE should become standard metrics for comparing forecasts across multiple series.

There are many more metrics proposed by various authors (see Table A.3), some of them tailored to specific use cases, therefore multiple error measures should always be used when comparing methods/models. Different measures also give different insight into possible problems that may lay in the problems. In this research, we mainly use: MAE, MAPE, MASE, and R^2.

Table A.3: Most used evaluation metrics. The following quantities are used:$e_t = y_t - f_t$, $p_t = ((y_t - f_t)/y_t)$, and $q_t = e_t/(\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|)$.

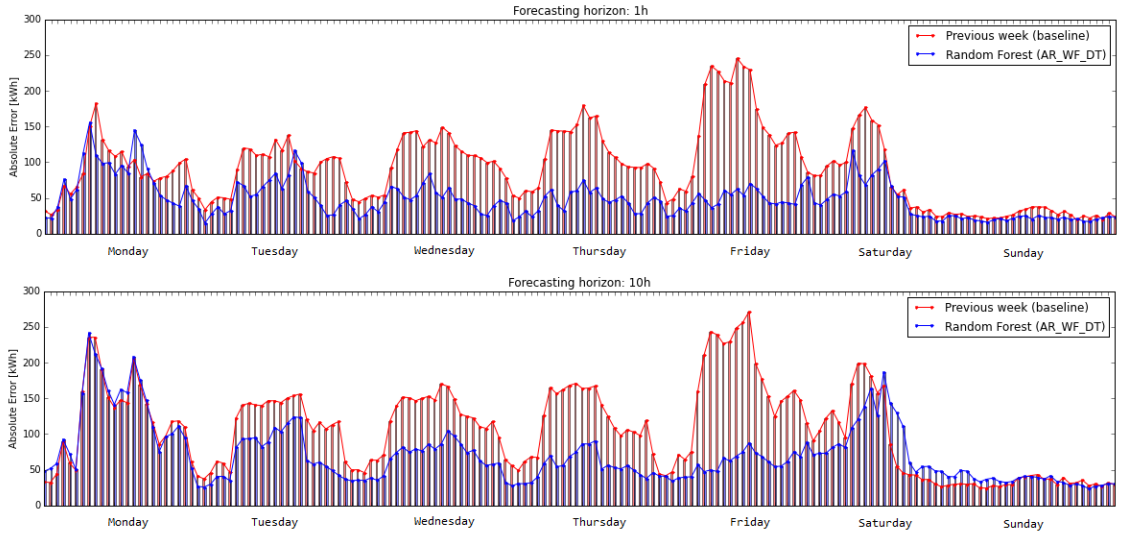
Abbr.	Name	Formula	Description
ME	Mean error	$\frac{1}{n}\sum_{t=1}^{n} e_t$	ME is likely to be small, as positive and negative errors tend to offset one another. This measure can only tell us whether a forecast bias exists in the model.
MAE	Mean absolute error	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$	MAE removes the original disadvantage of the ME with the introduction of the absolute value.
MSE	Mean squared error	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$	MSE is also not strained with positive/negative error compensation like MAE, but it is a bit more difficult to interpret.
RMSE	Root mean squared error	$\sqrt{MSE}$	Square root of MSE. Frequently used measure for numerical predictions. Compared to the similar MAE, RMSE amplifies and severely punishes large errors. Should not be used for comparisons across series.
MAPE	Mean absolute percentage error	$\frac{1}{n}\sum_{t=1}^{n} \lvert p_t \rvert$	Percentage error. Advantage of being scale-independent, which means that it can be used for comparisons of models that use different time series. It has known problems, such as zero values in stream will cause errors.
sMAPE	Symmetric mean absolute percentage error	$2\sum_{t=1}^{n} \frac{\lvert e_t \rvert}{f_t + y_t}$	This alternative to MAPE is limited to 2, but behaves better with low value items in the series. Low items can otherwise have infinitely high error rates that skew the overall error rate.
MASE	Mean absolute scaled error	$\frac{1}{n}\sum_{t=1}^{n} q_t$	Scale-free metric that can be used to compare forecast methods on a single series and between series. Uses naïve model for reference. It is also suitable for series with zero values. If score is lower than 1, the model is better than the baseline.
$\boldsymbol{R^2}$	Coefficient of determination	$\frac{\sum_{t=1}^{n}(\hat{x}(k) - \bar{x}(t))^2}{\sum_{t=1}^{n}(x(k) - \bar{x}(t))^2}$	Measure of fitness that basically tells how much better from the normal overall average is the model. Values range from 0-1. The closer to 1, the better the fit. Values can also be negative; it means that the model is worse than the simple average.

A.3 Error Analysis of STLF Predictions

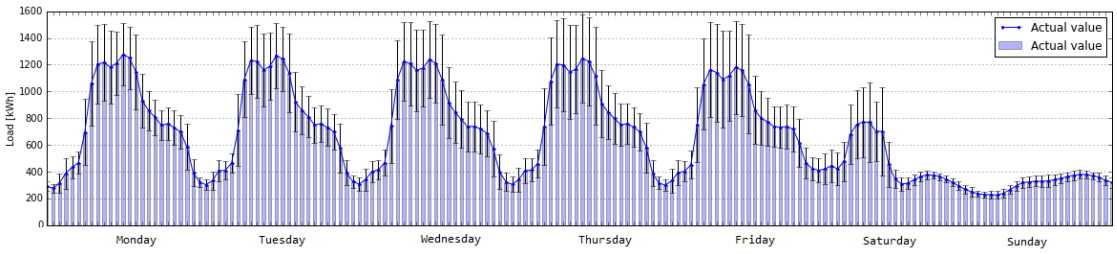
The analysis was done for two different forecasting horizons: 1h and 10h, for two estimators: Previous week (baseline), and Random Forest (using AR_WF_DT dataset). Figure A.2a presents the absolute errors for node *id12041022*, averaged by day of week and hours of day. It is clear that forecasts from the Random Forest model are significantly better than both baseline estimators (for both prediction horizons).

We can also see that that the baseline estimator "Previous Week" has the biggest errors on Fridays and Saturdays, but Random Forest has the biggest error on Mondays and Saturday (for both forecasting horizons). For the "Previous Week" estimator, the reason for this is that the variance of load demand is the highest on Fridays (due to different working hours, extended weekends) and Saturdays (again due to different working hours), as it can be seen in Figure A.2b. But for the Random Forest model, the most probable explanation for the biggest errors on Mondays and Saturdays is the fact that the load change is bigger than usual from Sundays to Mondays, and from Fridays to Saturdays (this can be seen in Figure A.2b and from exploratory analysis section in 3.3.2). Meaning that if we are making 10h-long predictions for Monday, we use the input measurements from Sunday, which is a lot different than Monday (the same goes for Saturday predictions).

In order to improve this matter, one possible solution would be to find an additional feature that would help the model to recognize these bigger changes. Another solution, which is more likely to be successful, is to build a separate model for each day of the week (or at least business and non-business days).



(a) Absolute errors, averaged by hour and day of the week



(b) Load demand, averaged by hour and day of the week

Figure A.2: Error analysis reveals the biggest errors for predictions made from business days (e.g. Friday) to non-business days (e.g. Saturday) and vice versa.

References

[1] U. Siebert, O. Alagoz, A. M. Bayoumi, B. Jahn, D. K. Owens, D. J. Cohen, and K. M. Kuntz, "State-transition modeling: A report of the ispor-smdm modeling good research practices task force–3," *Medical Decision Making*, vol. 32, no. 5, pp. 690–700, 2012.

[2] V. Etter, M. Kafsi, E. Kazemi, M. Grossglauser, and P. Thiran, "Where to go from here? mobility prediction from instantaneous information," *Pervasive and Mobile Computing*, vol. 9, no. 6, pp. 784–797, 2013.

[3] D. Brockmann, L. Hufnagel, and T. Geisel, "The scaling laws of human travel," *Nature*, vol. 439, no. 7075, pp. 462–465, 2006.

[4] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[5] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE/ACM transactions on networking (TON)*, vol. 19, no. 3, pp. 630–643, 2011.

[6] K. Zhao, M. Musolesi, P. Hui, W. Rao, and S. Tarkoma, "Explaining the power-law distribution of human mobility through transportation modality decomposition," *Scientific reports*, vol. 5, 2015.

[7] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[8] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, 2003.

[9] R. Wang, F. Chen, Z. Chen, T. Li, G. Harari, S. Tignor, X. Zhou, D. Ben-Zeev, and A. T. Campbell, "Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 2014, pp. 3–14.

[10] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating next-cell predictors with extensive wi-fi mobility data," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1633–1649, 2006.

[11] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain, "Predictability of wlan mobility and its effects on bandwidth provisioning.," in *INFOCOM*, 2006.

[12] J. G. Cleary and W. J. Teahan, "Unbounded length contexts for ppm," *The Computer Journal*, vol. 40, no. 2 and 3, pp. 67–75, 1997.

[13] P. Jacquet, W. Szpankowski, and I. Apostol, "A universal predictor based on pattern matching," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1462–1472, 2002.

[14] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Show me how you move and i will tell you who you are," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, ACM, 2010, pp. 34–41.

[15] ——, "Next place prediction using mobility markov chains," in *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, ACM, 2012, p. 3.

[16] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "Nextplace: A spatio-temporal prediction framework for pervasive systems," in *International Conference on Pervasive Computing*, Springer, 2011, pp. 152–169.

[17] H. Kantz and T. Schreiber, *Nonlinear time series analysis*. Cambridge university press, 2004, vol. 7.

[18] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, *CRAWDAD dataset epfl/mobility (v. 2009-02-24)*, Downloaded from `http://crawdad.org/epfl/mobility/20090224`, Feb. 2009. DOI: `10.15783/C7J010`.

[19] M. Musolesi, K. Fodor, M. Piraccini, A. Corradi, and A. Campbell, *CRAWDAD dataset dartmouth/cenceme (v. 2008-08-13)*, Downloaded from `http://crawdad.org/dartmouth/cenceme/20080813`, Aug. 2008. DOI: `10.15783/C76P4X`.

[20] M. Lenczner and A. G. Hoen, *CRAWDAD dataset ilesansfil/wifidog (v. 2015-11-06)*, Downloaded from `http://crawdad.org/ilesansfil/wifidog/20151106`, Nov. 2015. DOI: `10.15783/C7H883`.

[21] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden markov models," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 2012, pp. 911–918.

[22] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory.," *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.

[23] J. K. Laurila, D. Gatica-Perez, I. Aad, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, M. Miettinen, *et al.*, "The mobile data challenge: Big data for mobile computing research," in *Pervasive Computing*, 2012.

[24] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila, "Towards rich mobile phone datasets: Lausanne data collection campaign," *Proc. ICPS, Berlin*, 2010.

[25] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2011, pp. 1082–1090.

[26] L. Wen, X. Shi-xiong, L. Feng, and Z. Lei, "Improving location prediction by exploring spatial-temporal-social ties," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[27] R. Yu, X. Xia, S. Liao, and X. Wang, "A location prediction algorithm with daily routines in location-based participatory sensing systems," *International Journal of Distributed Sensor Networks*, vol. 2015, p. 6, 2015.

[28] S.-B. Cho, "Exploiting machine learning techniques for location recognition and prediction with smartphone logs," *Neurocomputing*, vol. 176, pp. 98–106, 2016.

[29] Q. Lv, Y. Qiao, N. Ansari, J. Liu, and J. Yang, "Big data driven hidden markov model based individual mobility prediction at points of interest," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 5204–5216, 2016.

[30] F. D. N. Neto, C. de Souza Baptista, and C. E. Campelo, "A user-personalized model for real time destination and route prediction," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2016, pp. 401–407.

[31] ——, "Combining markov model and prediction by partial matching compression technique for route and destination prediction," *Knowledge-Based Systems*, vol. 154, pp. 81–92, 2018.

[32] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[33] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," *Artificial Intelligence*, vol. 259, pp. 147–166, 2018.

[34] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting places from traces of locations," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 3, pp. 58–68, 2005.

[35] M. Senožetnik, L. Bradeško, B. Kažič, D. Mladenic, and T. Šubic, "Spatio-temporal clustering methods," in *Proceedings of the 19th International Multiconference Information Society - IS 2016*, Ljubljana, Slovenia, 2016, pp. 33–36.

[36] M. Lv, L. Chen, Z. Xu, Y. Li, and G. Chen, "The discovery of personally semantic places based on trajectory data mining," *Neurocomputing*, vol. 173, pp. 1142–1153, 2016.

[37] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[38] B. Kažič, J. Rupnik, P. Škraba, L. Bradeško, and D. Mladenič, "Predicting users' mobility using monte carlo simulations," *IEEE Access*, vol. 5, pp. 27 400–27 420, 2017, COBISS category: *1A1 (Z, A', A1/2)*, JCR impact factor: 3.557. DOI: 10.1109/ACCESS.2017.2768125.

[39] D. M. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[40] X. Zhu, *Knowledge discovery and data mining: Challenges and realities: Challenges and realities*. Igi Global, 2007, pp. 118–119.

[41] J. Urbančič, V. Pejovič, and D. Mladenić, "Transportation mode detection using random forest," *Proceedings of SIKDD 2018*, 2018.

[42] A. De Mauro, M. Greco, and M. Grimaldi, "A formal definition of big data based on its essential features," *Library Review*, vol. 65, no. 3, pp. 122–135, 2016.

[43] B. Kažič and J. Rupnik, "Sensor-based single-user activity recognition," in *Proceedings of the 16th International Multiconference Information Society - IS 2013*, vol. A, Ljubljana, Slovenia, 2013, pp. 180–183.

[44] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[45] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European symposium on artificial neural networks (esann)*, 2016.

[46] M. Kholghi and M. Keyvanpour, "An analytical framework for data stream mining techniques based on challenges and requirements," *ArXiv preprint arXiv:1105.1950*, 2011.

[47] S. Fong, Y. Zhuang, R. Wong, and S. Mohammed, "A scalable data stream mining methodology: Stream-based holistic analytics and reasoning in parallel," in *2014 2nd International Symposium on Computational and Business Intelligence*, IEEE, 2014, pp. 110–115.

[48] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604, 2010.

[49] B. Fortuna, J. Rupnik, J. Brank, C. Fortuna, V. Jovanoski, M. Karlovčec, B. Kažič, K. Kenda, G. Leban, A. Muhič, B. Novak, J. Novljan, M. Papler, L. Rei, B. Sovdat, L. Stopar, M. Grobelnik, and D. Mladenić, "Qminer: Data analytics platform for processing streams of structured and unstructured data," in *Software engineering for machine learning workshop, Neural Information Processing Systems (NIPS)*, 2014.

[50] M. Venkatasubramanian and K. Tomsovic, "Power system analysis," in *The Electrical Engineering Handbook*, Elsevier, 2005, pp. 761–778.

[51] K. Zhang, X. R. Li, and Y. Zhu, "Optimal update with out-of-sequence measurements," *IEEE Transactions on Signal Processing*, vol. 53, no. 6, pp. 1992–2004, 2005.

[52] A. Akbar, G. Kousiouris, H. Pervaiz, J. Sancho, P. Ta-Shma, F. Carrez, and K. Moessner, "Real-time probabilistic data fusion for large-scale iot applications," *IEEE Access*, vol. 6, pp. 10 015–10 027, 2018.

[53] D. Lahat, T. Adali, and C. Jutten, "Multimodal data fusion: An overview of methods, challenges, and prospects," *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1449–1477, 2015.

[54] B. Khaleghi, A. Khamis, and F. Karray, "Multisensor data fusion: A data-centric review of the state of the art and overview of emerging trends," in *Multisensor Data Fusion*, CRC Press, 2015, pp. 38–57.

[55] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, vol. 1, 2000, pp. 288–293.

[56] A. Amini, T. Y. Wah, and H. Saboohi, "On density-based data streams clustering algorithms: A survey," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 116–141, 2014.

[57] E. Harrington, R. Herbrich, J. Kivinen, J. Platt, and R. C. Williamson, "Online bayes point machines," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2003, pp. 241–252.

[58] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

[59] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in neural information processing systems*, 2001, pp. 409–415.

[60] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth, "Using and combining predictors that specialize," in *In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, Citeseer, 1997.

[61] W. Yi, F. Teng, and J. Xu, "Noval stream data mining framework under the background of big data," *Cybernetics and Information Technologies*, vol. 16, no. 5, pp. 69–77, 2016.

[62] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Kdd*, vol. 2, 2000, p. 4.

[63] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*, Springer, 1994, pp. 409–426.

[64] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data mining and knowledge discovery*, vol. 23, no. 1, pp. 128–168, 2011.

[65] A. Osojnik, P. Panov, and S. Džeroski, "Multi-label classification via multi-target regression on data streams," *Machine Learning*, vol. 106, no. 6, pp. 745–770, 2017.

[66] C. Manapragada, G. I. Webb, and M. Salehi, "Extremely fast decision tree," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018, pp. 1953–1962.

[67] N. Kourtellis, G. D. F. Morales, A. Bifet, and A. Murdopo, "Vht: Vertical hoeffding tree," in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 915–922.

[68] N. Marz and J. Warren, *Big data: Principles and best practices of scalable real-time data systems*. New York; Manning Publications Co., 2015.

[69] M.-R. Bouguelia, A. Karlsson, S. Pashami, S. Nowaczyk, and A. Holst, "Mode tracking using multiple data streams," *Information Fusion*, vol. 43, pp. 33–46, 2018.

[70] D. Q. Tu, A. Kayes, W. Rahayu, and K. Nguyen, "Isdi: A new window-based framework for integrating iot streaming data from multiple sources," in *International Conference on Advanced Information Networking and Applications*, Springer, 2019, pp. 498–511.

[71] T. Hong and S. Fan, "Probabilistic electric load forecasting: A tutorial review," *International Journal of Forecasting*, vol. 32, no. 3, pp. 914–938, 2016.

[72] K. Zor, O. Timur, and A. Teke, "A state-of-the-art review of artificial intelligence techniques for short-term electric load forecasting," in *2017 6th International Youth Conference on Energy (IYCE)*, IEEE, 2017, pp. 1–7.

[73] S. R. Khuntia, J. L. Rueda, and M. A. van der Meijden, "Forecasting the load of electrical power systems in mid-and long-term horizons: A review," *IET Generation, Transmission & Distribution*, vol. 10, no. 16, pp. 3971–3977, 2016.

[74] F. Ardakani and M. Ardehali, "Long-term electrical energy consumption forecasting for developing and developed economies based on different optimized models and historical data types," *Energy*, vol. 65, pp. 452–461, 2014.

[75] P. Sutthichaimethee and K. Kubaha, "The efficiency of long-term forecasting model on final energy consumption in thailand's petroleum industries sector: Enriching the lt-arimaxs model under a sustainability policy," *Energies*, vol. 11, no. 8, p. 2063, 2018.

[76] T. Al-Saba and I. El-Amin, "Artificial neural networks as applied to long-term demand forecasting," *Artificial Intelligence in Engineering*, vol. 13, no. 2, pp. 189–197, 1999.

[77] A. Bruhns, G. Deurveilher, and J.-S. Roy, "A non linear regression model for mid-term load forecasting and improvements in seasonality," in *Proceedings of the 15th Power Systems Computation Conference*, Citeseer, 2005, pp. 22–26.

[78] D. H. Vu, K. M. Muttaqi, and A. Agalgaonkar, "A variance inflation factor and backward elimination based robust regression model for forecasting monthly electricity demand using climatic variables," *Applied Energy*, vol. 140, pp. 385–394, 2015.

[79] G. Tsekouras, E. Dialynas, N. Hatziargyriou, and S. Kavatza, "A non-linear multivariable regression model for midterm energy forecasting of power systems," *Electric Power Systems Research*, vol. 77, no. 12, pp. 1560–1568, 2007.

[80] N. Abu-Shikhah, F. Elkarmi, and O. M. Aloquili, "Medium-term electric load forecasting using multivariable linear and non-linear regression," *Smart Grid and Renewable Energy*, vol. 2, no. 02, p. 126, 2011.

[81] A. Ahmad, M. Hassan, M. Abdullah, H. Rahman, F. Hussin, H. Abdullah, and R. Saidur, "A review on applications of ann and svm for building electrical energy consumption forecasting," *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 102–109, 2014.

[82] J. Li, J. Liu, J. Wang, *et al.*, "Mid-long term load forecasting based on simulated annealing and svm algorithm," *Proceedings of the CSEE*, vol. 31, no. 16, pp. 63–66, 2011.

[83] M. Ghiassi, D. K. Zimbra, and H. Saidane, "Medium term system load forecasting with a dynamic artificial neural network model," *Electric power systems research*, vol. 76, no. 5, pp. 302–316, 2006.

[84] L. Han, Y. Peng, Y. Li, B. Yong, Q. Zhou, and L. Shu, "Enhanced deep networks for short-term and medium-term load forecasting," *IEEE Access*, vol. 7, pp. 4045–4055, 2018.

[85] H. Hahn, S. Meyer-Nieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," *European journal of operational research*, vol. 199, no. 3, pp. 902–907, 2009.

[86] A. Muñoz, E. F. Sánchez-Úbeda, A. Cruz, and J. Marın, "Short-term forecasting in power systems: A guided tour," in *Handbook of power systems II*, Springer, 2010, pp. 129–160.

[87] J. R. Cancelo, A. Espasa, and R. Grafe, "Forecasting the electricity load from one day to one week ahead for the spanish system operator," *International Journal of forecasting*, vol. 24, no. 4, pp. 588–602, 2008.

[88] E. Feinberg and D. Genethliou, *Load forecasting, applied mathematics for restructured electric power systems*, 2006.

[89] J. N. Fidalgo and M. A. Matos, "Forecasting portugal global load with artificial neural networks," in *International Conference on Artificial Neural Networks*, Springer, 2007, pp. 728–737.

[90] J. W. Taylor and R. Buizza, "Using weather ensemble predictions in electricity demand forecasting," *International Journal of Forecasting*, vol. 19, no. 1, pp. 57–70, 2003.

[91] J. W. Taylor, "An evaluation of methods for very short-term load forecasting using minute-by-minute british data," *International Journal of Forecasting*, vol. 24, no. 4, pp. 645–658, 2008.

[92] M. Alamaniotis, A. Ikonomopoulos, and L. H. Tsoukalas, "Evolutionary multiobjective optimization of kernel-based very-short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1477–1484, 2012.

[93] C. Guan, P. B. Luh, L. D. Michel, Y. Wang, and P. B. Friedland, "Very short-term load forecasting: Wavelet neural networks with data pre-filtering," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 30–41, 2012.

[94] M. Espinoza, C. Joye, R. Belmans, and B. De Moor, "Short-term load forecasting, profile identification, and customer segmentation: A methodology based on periodic time series," *IEEE Transactions on Power Systems*, vol. 20, no. 3, pp. 1622–1630, 2005.

[95] E. Kyriakides and M. Polycarpou, "Short term electric load forecasting: A tutorial," in *Trends in Neural Computation*, Springer, 2007, pp. 391–418.

[96] R. Ramanathan, R. Engle, C. W. Granger, F. Vahid-Araghi, and C. Brace, "Short-run forecasts of electricity loads and peaks," *International journal of forecasting*, vol. 13, no. 2, pp. 161–174, 1997.

[97] N. Charlton and C. Singleton, "A refined parametric model for short term load forecasting," *International Journal of Forecasting*, vol. 30, no. 2, pp. 364–368, 2014.

[98] P. Mirowski, S. Chen, T. K. Ho, and C.-N. Yu, "Demand forecasting in smart grids," *Bell Labs technical journal*, vol. 18, no. 4, pp. 135–158, 2014.

[99] B.-J. Chen, M.-W. Chang, *et al.*, "Load forecasting using support vector machines: A study on eunite competition 2001," *IEEE transactions on power systems*, vol. 19, no. 4, pp. 1821–1830, 2004.

[100] H. Hippert, D. Bunn, and R. Souza, "Large neural networks for electricity load forecasting: Are they overfitted?" *International Journal of forecasting*, vol. 21, no. 3, pp. 425–434, 2005.

[101] A. Almalaq and G. Edwards, "A review of deep learning methods applied on load forecasting," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2017, pp. 511–516.

[102] T. Palmer, G. Shutts, R. Hagedorn, F. Doblas-Reyes, T. Jung, and M. Leutbecher, "Representing model uncertainty in weather and climate prediction," *Annu. Rev. Earth Planet. Sci.*, vol. 33, pp. 163–193, 2005.

[103] M.-C. Tan, S. C. Wong, J.-M. Xu, Z.-R. Guan, and P. Zhang, "An aggregation approach to short-term traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 60–69, 2009.

[104] H. R. Kirby, S. M. Watson, and M. S. Dougherty, "Should we use neural networks or statistical models for short-term motorway traffic forecasting?" *International Journal of Forecasting*, vol. 13, no. 1, pp. 43–50, 1997.

[105] C. Shearer, "The crisp-dm model: The new blueprint for data mining," *Journal of data warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[106] K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the internet of things," *Sensors*, vol. 19, no. 8, p. 1955, 2019, COBISS category: *1A1 (Z, A', A1/2)*, JCR impact factor: 3.275. DOI: 10.3390/s19081955.

[107] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[108] B. Kažič, D. Mladenić, and A. Košmerlj, "Traffic flow prediction from loop counter sensor data using machine learning methods," in *VEHITS 2015: Proceedings of the 1st International Conference on Vehicle Technology and Intelligent Systems*, Lisbon, Portugal, 2015, pp. 119–127.

[109] B. Kažič, D. Mladenić, and L. Bradeško, "Complex event detection and prediction in traffic," in *Proceedings of the 17th International Multiconference Information Society - IS 2014*, vol. A, Ljubljana, Slovenia, 2014, pp. 18–21.

[110] B. Kažič, K. Kenda, and D. Mladenić, "Load forecasting for smart electricity distribution in sunseed project," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018.

[111] B. Kažič, M. Škrjanc, P. Škraba, D. Mladenić, U. Kuhar, A. Medved, K. Alič, R. Novak, A. Švigelj, and J. JurŠe, "Pseudo-measurements based on the forecasted smart meter consumption for state-estimation: Testbed deployment case study," in *EMENDER 2015 - energy managemENT Data ElaboRation*, Lisbon, Slovenia, 2015, p. 6.

[112] R. Sevlian and R. Rajagopal, "Short term electricity load forecasting on varying levels of aggregation," *ArXiv preprint arXiv:1404.0058*, 2014.

[113] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro, *et al.*, "Application of high-dimensional feature selection: Evaluation for genomic prediction in man," *Scientific reports*, vol. 5, p. 10 312, 2015.

[114] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[115] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. RĆ1/4ckstieĆ, and J. Schmidhuber, "Pybrain," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 743–746, 2010.

[116] R. J. Hyndman *et al.*, "Another look at forecast-accuracy metrics for intermittent demand," *Foresight: The International Journal of Applied Forecasting*, vol. 4, no. 4, pp. 43–46, 2006.

[117] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy and Buildings*, vol. 49, pp. 591–603, 2012.

[118] H. Ziekow, C. Goebel, J. Struker, and H.-A. Jacobsen, "The potential of smart home sensors in forecasting household electricity demand," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, IEEE, 2013, pp. 229–234.

[119] R. P. Singh, P. X. Gao, and D. J. Lizotte, "On hourly home peak load prediction," in *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, IEEE, 2012, pp. 163–168.

[120] M. Ghofrani, M. Hassanzadeh, M. Etezadi-Amoli, and M. S. Fadali, "Smart meter based short-term load forecasting for residential customers," in *North American Power Symposium (NAPS), 2011*, IEEE, 2011, pp. 1–5.

[121] G. Adepoju, S. Ogunjuyigbe, and K. Alawode, "Application of neural network to load forecasting in nigerian electrical power system," *The Pacific Journal of Science and Technology*, vol. 8, no. 1, pp. 68–72, 2007.

[122] D. Benaouda, F. Murtagh, J.-L. Starck, and O. Renaud, "Wavelet-based nonlinear multiscale decomposition model for electricity load forecasting," *Neurocomputing*, vol. 70, no. 1-3, pp. 139–154, 2006.

[123] A. Al-Shareef, E. Mohamed, and E. Al-Judaibi, "One hour ahead load forecasting using artificial neural network for the western area of saudi arabia," *International Journal of Electrical Systems Science and Engineering*, vol. 1, no. 1, pp. 35–40, 2008.

[124] I. Drezga and S. Rahman, "Short-term load forecasting with local ann predictors," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 844–850, 1999.

[125] K. Lee, Y. Cha, and J. Park, "Short-term load forecasting using an artificial neural network," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 124–132, 1992.

[126] T. Senjyu, H. Takara, K. Uezato, and T. Funabashi, "One-hour-ahead load forecasting using neural network," *IEEE Transactions on power systems*, vol. 17, no. 1, pp. 113–118, 2002.

[127] K. Kenda, B. Kažič, L. Stopar, B. Fortuna, J. Rupnik, M. Škrjanc, and D. Mladenić, "Data fusion framework for streaming heterogeneous data sources," *Journal of Computer Science and Technology (submitted)*, 2018.

[128] G. Leban, B. Fortuna, J. Brank, and M. Grobelnik, "Event registry: Learning about world events from news," in *Proceedings of the 23rd International Conference on World Wide Web*, ACM, 2014, pp. 107–110.

Bibliography

Publications Related to the Thesis

Journal Articles

B. Kažič, J. Rupnik, P. Škraba, L. Bradeško, and D. Mladenič, "Predicting users' mobility using monte carlo simulations," *IEEE Access*, vol. 5, pp. 27 400–27 420, 2017, COBISS category: *1A1 (Z, A', A1/2)*, JCR impact factor: 3.557. DOI: 10.1109/ACCESS.2017.2768125.

K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the internet of things," *Sensors*, vol. 19, no. 8, p. 1955, 2019, COBISS category: *1A1 (Z, A', A1/2)*, JCR impact factor: 3.275. DOI: 10.3390/s19081955.

Conference Papers

B. Kažič, K. Kenda, and D. Mladenić, "Load forecasting for smart electricity distribution in sunseed project," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018.

B. Fortuna, J. Rupnik, J. Brank, C. Fortuna, V. Jovanoski, M. Karlovčec, B. Kažič, K. Kenda, G. Leban, A. Muhič, B. Novak, J. Novljan, M. Papler, L. Rei, B. Sovdat, L. Stopar, M. Grobelnik, and D. Mladenić, "Qminer: Data analytics platform for processing streams of structured and unstructured data," in *Software engineering for machine learning workshop, Neural Information Processing Systems (NIPS)*, 2014.

M. Senožetnik, L. Bradeško, B. Kažič, D. Mladenic, and T. Šubic, "Spatio-temporal clustering methods," in *Proceedings of the 19th International Multiconference Information Society - IS 2016*, Ljubljana, Slovenia, 2016, pp. 33–36.

B. Kažič, M. Škrjanc, P. Škraba, D. Mladenić, U. Kuhar, A. Medved, K. Alič, R. Novak, A. Švigelj, and J. JurŠe, "Pseudo-measurements based on the forecasted smart meter consumption for state-estimation: Testbed deployment case study," in *EMENDER 2015 - energy managemENT Data ElaboRation*, Lisbon, Slovenia, 2015, p. 6.

B. Kažič, D. Mladenić, and A. Košmerlj, "Traffic flow prediction from loop counter sensor data using machine learning methods," in *VEHITS 2015: Proceedings of the 1st International Conference on Vehicle Technology and Intelligent Systems*, Lisbon, Portugal, 2015, pp. 119–127.

B. Kažič, D. Mladenić, and L. Bradeško, "Complex event detection and prediction in traffic," in *Proceedings of the 17th International Multiconference Information Society - IS 2014*, vol. A, Ljubljana, Slovenia, 2014, pp. 18–21.

B. Kažič and J. Rupnik, "Sensor-based single-user activity recognition," in *Proceedings of the 16th International Multiconference Information Society - IS 2013*, vol. A, Ljubljana, Slovenia, 2013, pp. 180–183.

Biography

Blaž Kažič is a Ph.D. candidate in Information and Communication Technologies at the Jožef Stefan International Postgraduate School. He earned his bachelor degree in Electrical Engineering from the University of Ljubljana. His research interests includes predictive analytics with machine learning methods, big data analytics, exploratory data mining and applying these techniques to real-world applications. His dissertation examines predictive modeling approaches in the field of smart mobility and smart grid sectors, using high-volume heterogeneous streaming time series data. During his Ph.D. studies he was employed as a researcher at the Artificial Intelligence Laboratory at the Jožef Stefan Institute and participated in a number of EU-founded projects, such as MobiS project (personalized mobility services using AI techniques), and SUNSEED project (Sustainable and robust networking for smart electricity distribution). During his study he also collaborated on various industrial projects with companies such as Kolektor Group d.o.o., Iskratel d.o.o., Adria Mobil d.o.o., BT Group plc, and Bloomberg L.P.