University of Zurich[UZH]

Department of Geography

# Incorporating Spatio-Temporal Context for Predicting the Next Place Using Neural Networks and Random Forests

GEO 511 Master's Thesis


**Author**

Urner Jorim

Matr.-No: 11-724-416

**Supervised by**

Dominik Bucher (dobucher@ethz.ch)[1] , Dr. David Jonietz (jonietzd@ethz.ch)[1], Prof. Dr. Martin Raubal (mraubal@ethz.ch)[1], Jing Yang (jing.yang@inf.ethz.ch)[2]

**Faculty representative**

Prof. Dr. Ross Purves


26.01.2018

Department of Geography, University of Zurich

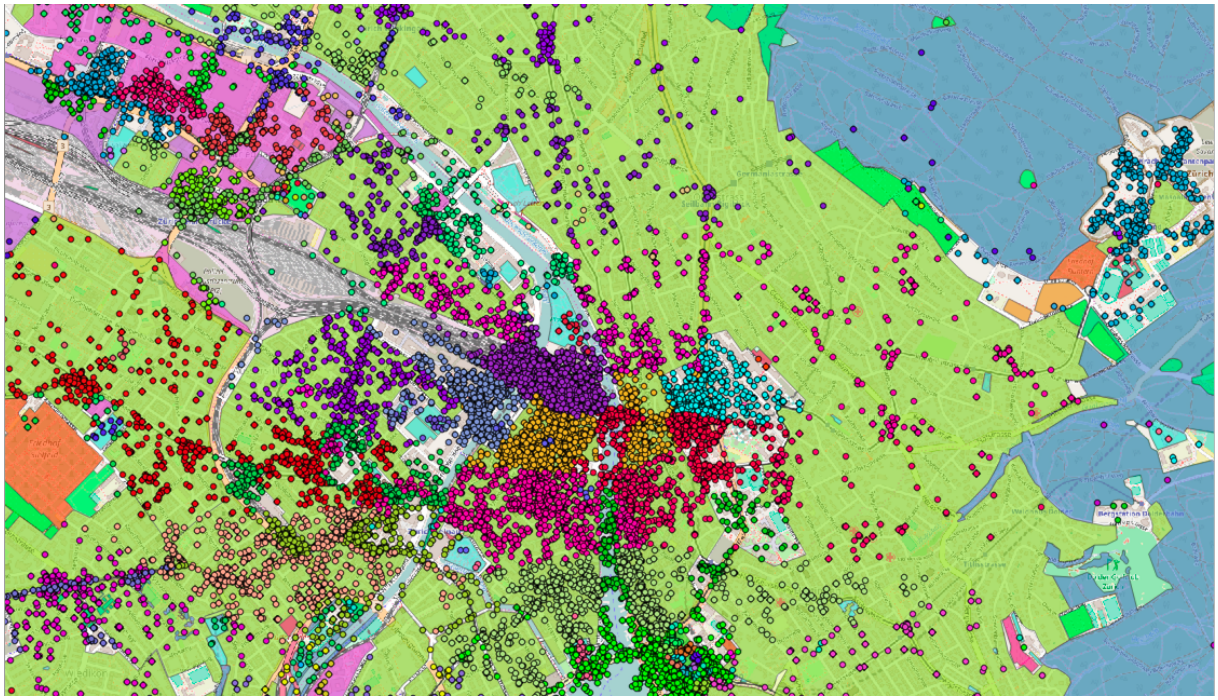

[1] Institute of Cartography and Geoinformation, Dept. of Civil, Environmental and Geomatic Engineering, ETH Zurich, Stefano-Franscini-Platz 5, CH-8093 Zurich, Switzerland

[2] Institute for Pervasive Computing, Departement of Computer Science, ETH Zurich, Universitätstrasse 6 CH-8092 Zurich, Switzerland

University of Zurich[UZH]
Department of Geography

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Incorporating Spatio-Temporal Context for Predicting the Next Place Using Neural Networks and Random Forests



GEO511: Master's Thesis

Submission: 26.01.2018, Department of Geography, University of Zurich

**Author: Jorim Urner**
Matr.-No: 11-724-416

**Supervisors:**
Dominik Bucher, ETH
Jing Yang, ETH,
Dr. David Jonietz, ETH
Prof. Dr. Martin Raubal, ETH

**Faculty Representative:**
Prof. Dr. Ross Purves, UZH

## Contacts

**Dominik Bucher**
Institute of Cartography and Geoinformation
Dept. of Civil, Environmental and Geomatic Engineering
ETH Zurich
Stefano-Franscini-Platz 5
CH-8093 Zürich, Switzerland
dobucher@ethz.ch

**Dr. David Jonietz**
Institute of Cartography and Geoinformation
Dept. of Civil, Environmental and Geomatic Engineering
ETH Zurich
Stefano-Franscini-Platz 5
CH-8093 Zürich, Switzerland
jonietzd@ethz.ch

**Prof. Dr. Ross Purves**
Geographic Information Science (GIS), Department of Geography
University of Zurich - Irchel
Winterthurerstrasse 190
CH-8057 Zurich, Switzerland
ross.purves@geo.uzh.ch

**Prof. Dr. Martin Raubal**
Institute of Cartography and Geoinformation
Dept. of Civil, Environmental and Geomatic Engineering
ETH Zurich
Stefano-Franscini-Platz 5
CH-8093 Zürich, Switzerland
mraubal@ethz.ch

**Jorim Urner**
Agnesstrasse 44
CH-8004 Zürich, Switzerland
jorim.urner@gmail.com

**Jing Yang**
Institute for Pervasive Computing
Departement of Computer Science
ETH Zurich
Universitaetstrasse 6
CH-8092 Zurich, Switzerland
jing.yang@inf.ethz.ch

# Acknowledgments

This thesis was only made possible with the help of several people. First, I would like to thank my supervisors Dominik Bucher, David Jonietz, Prof. Ross Purves, Prof. Martin Raubal and Jing Yang for their valuable inputs and their assistance. Second, I want to thank the team of GoEco! for providing me with their data and last but not least, the Institute of Cartography and Geoinformation at the ETH for allowing me to use their server.

Furthermore, I thank all of my friends at University for the nice and challenging time we have spent together during the last six years. A special thank you goes to David Hanimann and Gina Steiger for their careful revision of my drafts and their critical feedback.

# Abstract

Mobility is fundamental to our daily lives. Not only does it shape our social relationships, but it also reflects our personal preferences. This is why it is important to study and understand human mobility. Thanks to the rise of smartphones in the last decade, the availability of mobility data has skyrocketed. The newly available data allows to tackle new challenges on the way to understanding human mobility.

This study explores how GPS data obtained from smartphone users can be used to predict the next place. While many proposed models in literature use user-specific predictors, this work uses one predictor for all users, hence, finds a suitable model structure that allows the model to output the places from all users. To handle a large number of possible outputs, a hierarchical structure is selected. Places are discretized into areas using rastering and clustering. With the areas obtained from these two methods, temporal and spatial features are extracted, which in turn are used to train a neural network and a random forest.

The results show that a user's destination can be predicted with an accuracy of 75.5 percent by an artificial neural network trained with the features created from the clusters. With the features extraced from rasters, the neural network, yields less accurate results and displays inferior performance due to the hierarchical prediction structure. The arbitrary division of space creates the modifiable area unit problem, which leads the model to be much more dependent on spatial features. The neural network trained with the clustered input, on the other side, values spatial and temporal features equally. Therefore it bases its decisions on more information and performs better.



**Keywords:** Human Mobility, Movement Patterns, Next Place Prediction, Feature Extraction, Neural Network, Random Forest

# Contents

# List of Figures

# List of Tables

# Abbreviations

**API** Application Programming Interface

**GPS** Global Positioning System

**LDA** Linear Discriminant Analysis

**MAUP** Modifiable Area Unit Problem

**MOT** Mode Of Transport

**NN** Artificial Neural Network

**PCA** Prinicipal Component Analysis

**RF** Random Forest

# 1 Introduction

Mobility is a crucial aspect of our daily lives. Places we visit shape our social relationships and reflect our personal preferences. The ability to predict the places a user will visit is therefore beneficial to numerous applications, ranging from forecasting the dynamics of crowds to improving the relevance of location-based recommendations (Etter et al., 2012).

The rapidly growing number of smartphone users across the world, has produced a massive amount of location data (Xu et al., 2016). This multidimensional source of data offers new possibilities to tackle established research problems on human mobility. Larger datasets and increasing processing power make it possible to use neural networks (NN) and other machine learning techniques approach those research problems.

## 1.1 Context and Motivation

Understanding human mobility has a multitude of potential applications. Thus it has been a longstanding subject in academic research (Noulas et al., 2012). One particular problem is the next place prediction, where the challenge consists of predicting the next location of a user given the current location or trajectory (Gomes et al., 2013).
With the growing ability to collect information, more temporal and spatial contextual information is collected, and the location prediction problem becomes feasible (Liu et al., 2016). Applications, ranging from predicting the spread of human and electronic viruses, to city planning and traffic warnings, depend on our ability to predict the next location of individuals (Song et al., 2010).
Mobile devices and location-based services generate a significant amount of mobile data (Gao et al., 2015), which typically consists of coordinates associated with a timestamp and additional attributes. In order to render the data applicable, it needs to be preprocessed. The challenge in the latter, is to preprocess the data in a useful manner understandable to the predictor. To achieve this, the raw data is used to create two types of input features. These are temporal and spatial ones, derived from GPS data. Temporal input features help the predictor to learn temporal visitation patterns of different places. Commonly used heuristics are the day of the week, the hour of the day or weekday/weekend differences because they display periodic behavior of a user (Noulas et al., 2012). As shown by Wang et al. (2012), periodicity is a good indicator to predict the next location. Spatial features, on the other side, help the predictor to understand the spatial relationship between the places and the user. A commonly used spatial feature is

the distance or the direction between the user's position and the places that can be predicted (Noulas et al., 2012).

There are different ways to predict the next place. Most of the time either the shape of the current trajectory is compared to the shape of historical trajectories (Xue et al., 2013) or historical counts of place visits are used to predict the next place visit (Etter et al., 2012).
To train a model, the output (also called label) needs to be calculated for each sample in the data. There are different ways to do this. While some researchers try to predict x and y coordinates directly, most discretize space into areas to predict an associated ID (Alvarez-Garcia et al., 2010). The advantage of predicting the coordinates directly is that places, where a user has never been can be predicted. However, it is difficult to train such a model because it does not account for any prior information on the distribution of the data (De Brébisson et al., 2015). Therefore, in the major part of literature, a set of predefined areas or places are used as output.

## 1.2 Problem Statement and Aim

This master's thesis aims to extract spatial and temporal features from trajectory data and use these features to predict the next place of a user with the help of an artificial neural network (NN) and a random forest (RF). NN have already proven to be useful in modeling sequences (Jain et al., 2016) but if they are adapted to model the spatial and temporal information better (Liu et al., 2016, Jain et al., 2016), such models will improve even more. A RF will be used as a comparison, because it has a high robustness for noisy data and it generalizes well (Criminisi, 2011).

There is plenty of research on next place prediction, but most of the models have a separately trained predictor for every user. The goal of this work is to train one predictor for all users. User-specific predictors are easier to train because they only need to fit patterns from one user. However, the benefit of an overall model could be that it can transfer its knowledge of all users to predict users with little data. Since the amount of possible outputs for such a model is immense, the model structure needs to be adjusted accordingly. Therefore, a hierarchical prediction structure is proposed.

In most studies, a new prediction model structure is introduced, which impedes a comparison of the performance to other models. They mostly perform well, but it is unclear which input features are of most importance to the predictor and if the predictor had performed better

with other input features. Since a hierarchical prediction structure is used, it will be possible to analyze the feature importance on different scales. Therefore, one goal is to find out the feature importance of different features and show which features should be used in which context.

A further aspect that is given attention to is the discretization of space. To get a finite set of outputs, space can be discretized using rastering or clustering (Khoroshevsky et al., 2017, Zheng, Huang, et al., 2017, Gambs et al., 2012). Both methods are employed in literature, but they have not been compared to each other in the context of movement prediction. To find the strengths and weaknesses of both approaches, two separate models are trained, one with features extracted from the raster, the other with features from the clusters.

## 1.3 Structure

The structure of the thesis follows the work stages that have been undertaken. Section 2 summarizes the related work in the area of next place prediction. The basics of NN and RF will be explained, and the research gap will be presented. The overall methodology and and the research objectives are defined in section 3. In section 4, the concepts are tested with real-life data. There, the architecture of the prediction system, the data preprocessing, the setting of the parameters and the extraction of the features is explained. The results of the experiment are evaluated in section 5. Finally, the discussion and the limits of the chosen approach will be shown in section 6.

# 2 Related Work

## 2.1 Predictability of Human Movement

The challenge faced in this study is to model and predict human mobility. While from a personal point of view, our movement pattern does rarely seem random, because we know our motivations behind it. However, for an outside observer, our movement often seems to be random and unpredictable. Therefore, human mobility models are mostly stochastic. To capture the degree of predictability, entropy is probably the most fundamental quantity, because it describes if there is a lack of order and therefore a lack of predictability in the data. Song et al. (2010) used tracking data from cell phone towers and found that with the combination of the empirically determined user entropy and Fano's inequality (relates the average information lost in a noisy channel to the probability of the categorization error), there is a potential average predictability in user mobility of 93%. They define predictability as information-theoretic upper bound that limits any next place prediction algorithm in predicting the next place based on historical data. Segmenting each week into 168 hourly intervals, they found that on average 70% of the time, the user's most visited location in that timeslot coincides with the user's actual location. This accuracy is much higher between noon and 1 pm, 6 pm and 7 pm, and in the night when users are supposed to be at home. This accuracy has a clear minima because it corresponds to transition periods. Interestingly, they could not find a statistically significant difference in predictability in gender or age groups as well as in urban and rural societies. (Song et al., 2010)
The potential average predictability is very high and will be very difficult to reach with current predictors. The predictability also depends on the data. Since Song et al. (2010) use data from cell phone towers, their movement trajectories are coarse and the places they predict are quite big areas. Thus, a user moving between two nearby places may never leave this area and the prediction would remain correct even though the place was changed. If GPS data is available, the prediction would be more complex, because there are many more locations that can be predicted. Therefore, the potential predictability for GPS data should be smaller.

## 2.2 Next Place Prediction

Different approaches have been used for next place prediction, and they all have their limitations. In many cases, different variations of Markov chains are used to master this task (Alvarez-Garcia et al., 2010, Gambs et al., 2012, Tran et al., 2012, Lu et al., 2013). The problem with Markov chains is that they are constructed based on a strong independence assumption among different factors, which limits its performance (Liu et al., 2016). Another conventional method is Tensor

Factorization. While it has been successfully used for time-aware recommendations (Xiong et al., 2010) and spatio-temporal information modeling (Liu et al., 2016), it faces the cold start problem in predicting future actions (Liu et al., 2016). Compared to Tensor factorization and Markov chains, NN show promising performances (Liu et al., 2016).

### 2.2.1 Predictors

The most commonly used model for next place prediction is a mathematical model known as the Markov model. This model has a set of states and transition probabilities between those states. At any point, there is an exact state and the transition probabilities of moving from this state to others. This is the basis of the model computation, which only depends on the starting state of the given transition (Pankin, 1987). The use of Markov chains for next place prediction is pretty straightforward. Places represent the states, and the movement between those places are the transitions. We can count each user's transitions, which means that it is possible to calculate transition probabilities between all places for every user. With transition probabilities, a transition matrix can be computed. Given the current place, the transition matrix can be used to find the next most likely destination. One typical approach is to partition space uniformly into grid cells or roads into segments and use the cells or segments as the states of the Markov model (Alvarez-Garcia et al., 2010, Xue et al., 2013).

If there is additional data that correlates with the states, such as movement direction or speed, a hidden Markov model can be used. The states of the hidden Markov model cannot be observed directly and are therefore called hidden variables. Each of these hidden variables has a set of observations, that describe them, and are used to calculate the transition probabilities. In the context of movement prediction, these observations could be the mode of transport, speed, or the time of the day.

Alvarez-Garcia et al. (2010) use a hidden Markov model for predicting the next destination given only the input data of a partial trip. A trip is defined as a GPS-trajectory between two stay points and a partial trip represents the beginning of such a trajectory. As states, they cluster the final points of each trip with a threshold of 200 meters, and only clusters visited more than three times are considered. In their dataset, they have three users with three to twelve destination clusters. As input, they extract support points from the GPS tracks, which are shortly after crossroads. These support points are much more helpful to figure out where the user will go. To test their model, they feed the hidden Markov model with 25%, 50%, 75% and 90% of the traveled trip. After 25% of the trip, the accuracy lies between 36.1% and 64.2%. After 90% of the trip, the accuracy rises over 71.8% and can reach 94.5%.

Cho (2016) use a similar approach. They extract intermediate locations by clustering the GPS trajectories. These intermediate locations are used to create a simplified GPS trajectory. Instead of just using spatial and temporal data, they also use contextual data recorded by the smatphone, namely, the acceleration, the magnetic field and the orientation, to compute the mode of transport, which is also used as input together with the GPS trajectory. Visited places are extracted with the g-means clustering algorithm as described in Hamerly et al. (2004). To predict the next place they use a hidden Markov chain.

Traditional prediction systems use the historical spatial trajectories to match the ongoing trajectory. If the ongoing and the historical trajectories are similar, the ongoing trajectory is very likely to have the same destination as the historical trajectory (Xue et al., 2013). Xue et al. (2013) developed a new approach to preprocess trajectories. They decompose the trajectories into sub-trajectories comprising two neighboring locations, and then connect the sub-trajectories into synthesized trajectories. This means they create all possible connections from the sub-trajecories. If the ongoing trajectory matches a synthesized trajectory, it can be used to predict the next place. From the ongoing trip, all reachable places are queried, and since the transition probability for each sub-trajectory is available, a Markov model can be used to quantify the probability for each location. According to the authors, with this method, the number of queries for which a destination can be predicted is exponentially increased compared to traditional trajectory matching, and it also runs by two orders of magnitudes faster than their baseline algorithm (Xue et al., 2013).
While Alvarez-Garcia et al. (2010) and Xue et al. (2013) compare the shape of the ongoing trajectory to historical trajectories, Gambs et al. (2012) use the visited places as states and calculate the transition probabilities between these states. The visited places are extracted by clustering all staypoints of a user, using the DJ-clustering algorithm, which is explained in Zhou et al. (2004). They use three different datasets and each user has between five and nine visited places. For the prediction, they use a Markov model that keeps track of the n previously visited places. However, according to the authors choosing $n > 2$ does not seem to bring a substantial improvement. The accuracy of their prediction model ranges from 70% to 95%.

Next to Markov models, traditional classification algorithms can be used for next place prediction. Given the current location of a user and some features based on his or her historical movement, a classifier can be used to predict the next location. A commonly utilized classifier is the neural network.

De Brébisson et al. (2015) won the Kaggle taxi trajectory prediction challenge[1] using neural networks. Their dataset contains data from 442 taxis running for an entire year. Each taxi ride has the taxi ID and the start time of the ride. If available, the client phone number is used as client ID; otherwise the taxi stand ID is used. As the GPS-located trajectories are of varying sizes, but the neural network requires a fixed and uniform input size, they only utilize the first five and the last five trajectory points as input, in addition to the start time, the client ID / taxi stand ID, and the taxi ID. Their goal was to predict a destination with two scalar values (longitude, latitude). They found that it was difficult to train such a model because it does not take into account any prior information on the distribution of the data. To tackle this problem, they clustered the destinations into 1000 clusters. The output of the NN associates a scalar value to each of these clusters. The higher this value for a cluster, the higher the probability of the taxi ride ending up in that cluster. Based on these values, they calculated the weighted average of the cluster centers to get a coordinate as output. Various network architectures including bidirectional recurrent NN and memory networks were tested, but the Kaggle challenge was won with a simple feed-forward NN.

For the Nokia mobile data challenge, Etter et al. (2012) propose user specific predictors, which learn from the users' mobility histories to predict the next location based on the current context. As input, they use the places (stay points) and extract temporal features from the start and the end time such as the day of the week of the start time and a boolean value indicating whether the end time is on the weekend. Their predictors are a dynamical Bayesian network, a gradient boosted decision tree and a NN. With an accuracy of 60.83%, the NN reaches the highest accuracy, but there is a high variance between the users. Since every predictor may make different errors and one predictor may fail on samples where another excels, they try four different blending strategies to improve their prediction further. Empirical tests showed improvements over the individual predictors.

Liu et al. (2016) propose a new model called Spatial-Temporal Recurrent Neural Network (ST-RNN). It models local temporal and spatial contexts in each layer with time-specific transition matrices for different time intervals and distance-specific transition matrices for different geographical distances. Their experimental results yield significant improvements over compared methods on two typical datasets, i.e. the Global Terrorism database (collection of 125'000 terroristic incidents) and the Gowalla dataset (data from a location-based social network).

[1] https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i, accessed: 2017-12-28

| Study | Goal | Input Features | Output features | Predictor | Spatial Scale |
|---|---|---|---|---|---|
| Alvarez-Garcia et al. (2010) | Predict the next location with trajectory matching. | Simplified GPS trajectory | 3 - 12 clusters per user | Hidden Markov model | Country |
| Cho (2016) | Predict the next location with trajectory matching. | Simplified GPS trajectory, discretized time / date features, transportation mode | 16 - 50 clusters per user | Hidden Markov model | City |
| Xue et al. (2013) | Splitting trajectories into sub-trajectories to create synthesized trajectories. Predict the next location by matching synthesized trajectories with the current. | 10-90% of the sub-trajectories of the current trip | Endpoints of synthesized trajectories | Sub-Trajectory Synthesis | City |
| Gambs et al. (2012) | Predict the next place by using the transition probabilities between places. | Transition probabilities between places | 5 and 9 clusters per user | Markov model | Country and City |
| De Brébisson et al. (2015) | Predict the next destination of a taxi based on the beginning of its trajectory | First five and the last five trajectory points, start time, the client ID/taxi stand ID, and the taxi ID | 1000 clusters | NN | City |
| Etter et al. (2012) | Predict the next location with different predictors and combine the predictors using blending strategies to improve the accuracy. | Current location, discretized time / date features and places | X places extracted from trajectories for every user | Bayesian network, gradient boosted decision tree and a NN | City |
| Liu et al. (2016) | Predict where user will go at a given time | Time and distance specific transition matrices | next check-in / next province | Spatial-Temporal Recurrent Neural Networks | |

*Table 1: Reviewed studies on next place prediction*

### 2.2.2 Input Data

A next location predictor needs informative input and a label/output for every sample. Regarding GPS coordinates, most places are unique. Therefore, space needs to be discretized. Space discretization can be done with clustering or grid cells (Alvarez-Garcia et al., 2010, Xue et al., 2013). This discretization helps to assign a class to every sample, which can be used as label. In many cases, clustering algorithms are applied with a certain threshold (e.g., maximum area) to find relevant areas (Alvarez-Garcia et al., 2010, Burbey et al., 2008). In other approaches, the data is discretized into a raster, and the cell IDs are used as labels (Khoroshevsky et al., 2017, Sneha et al., 2014, Lin et al., 2012).

Many papers do not consider all stay points but exclude irrelevant and random ones such as a point where the user only went once and a point where a user simply stood still for a while due to the lack of orientation. To remove these stay points, thresholds such as the minimum stay time, minimum visits per area or just the top n most visited areas are used (Alvarez-Garcia et al., 2010, Gomes et al., 2013). This clearly reduces the complexity of the prediction task. If only places which have been visited regularly need to be predicted, it is easier to extract patterns because it is known what the trajectories to each place look like and at what times a user typically goes there.

After discretizing the stay points into areas, features should be chosen to describe the areas. The predictor needs contextual information about every area to predict the next area. Spatial and temporal context can be extracted from GPS logs. Further information can be incorporated using other sensors of a smartphone to get the acceleration, orientation or magnetic field (Cho, 2016). The input can be further enriched by adding other contextual information such as the weather (Hoang et al., 2016) and social media check-ins (Jing, 2016, Gunduz et al., 2013). Also, work and home locations, entertainement places and other points of interest can be extracted for each user (Yuan et al., 2010, Siła-Nowicka et al., 2015, Zheng, Zhang, et al., 2009).

With the spatial context, there are two different approaches to next location prediction. The first approach uses the shape of the ongoing trajectory, to find similarly shaped trajectories from the past. The destination where the past trajectories ended up will be predicted (Xue et al., 2013). The second approach uses past counts of where the user went from the location where he or she currently is, to predict the next location (Etter et al., 2012). In the first approach, the shape of the ongoing trajectory is analyzed to make a prediction. In the second approach, they find places where the user usually goes after the current stay point and calculate the visiting

probabilities to those places.

Additionally, the temporal context can be used to create those counts. Song et al. (2010) create for each week 168 time slots and extract the most visited place for every time slot. Liu et al. (2016) use time windows with different intervals as input for their recurrent NN. In their experiment, a time interval of 6 hours performs best. Li et al. (2015) create a three-dimensional array, where the first two dimensions represent the counts of frequency of visitation to the location, and the third dimension represents the time. Others use specific temporal attributes such as the day of the week, the starting and ending hour, weekend or weekday and hour of the week as temporal features (Etter et al., 2012, Gao et al., 2015, Baumann et al., 2013). In some cases, the raw values are just added to the input (Etter et al., 2012). For instance, the day of the week is encoded with a number between 1 and 7. In other cases, the number of visits in an area within those time periods are counted and inputted. (Noulas et al., 2012, Gao et al., 2015).

## 2.3 Artificial Neural Networks

The basic idea of a NN is to simulate a lot of interconnected brain cells in a very simplified way, to recognize patterns and classify data (Maind et al., 2014). A typical NN consists of a dozen to millions of artificial neurons arranged in a series of layers. Each of these neurons is connected to the other neurons in the layers on either side.

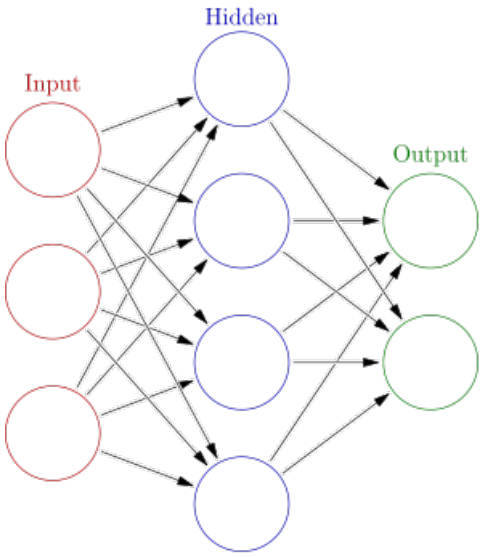



*Figure 2.1: A fully connected NN with one hidden layer*[2]*. It consists of input units (red), hidden units (blue) and output units (green). In every NN there is one layer with input units and one with output units. The number of hidden layers and units per layer can be changed. The connections between those units are associated with weights, which are represented by a number that is gradually adjusted during the training phase.*

[2]source: http://www.languageconnections.com/blog/neural-machine-translation/, accessed: 2018-4-1

Roughly speaking, a neural network implements a nonlinear mapping of *u=G(x)*. The mapping function $G$ is established during the training phase where the network learns to correctly associate input patterns $x$ to output patterns $u$ (Bebis et al., 1994). During the training phase, the input is fed into the input neurons. The input is multiplied by the weight of each connection and then added up in every hidden unit. This sum is then passed through the activation function which decides what the output should be. Usually, the number of output neurons is the same as the number of classes and the output neuron with the highest number will be the prediction.
Most recent NN usually use rectified linear units (ReLUs) as the activation function. A ReLU outputs zero if the input is less than or equal to zero otherwise it outputs the raw input (*f(x) = max(x, 0)*). The output will then be used as the input for the neurons in the next layer. The formula for a neuron is as follows:

$$y = f(\sum_{i=1}^{n} w_i x_i + b)$$

Where:

$w_i$ : $i^{th}$ weight

$x_i$ : $i^{th}$ input

b : bias

f : activation function

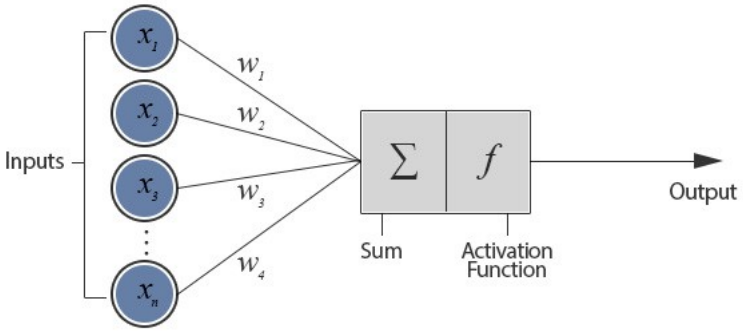



*Figure 2.2: A model of a neuron*[3]*. The inputs are multiplied by the weights and summed up. The sum is put into the activation function, which produces the output.*

A process called backpropagation learning is applied for learning during the training phase. The ground truth label l is compared to the output y using the formula:
*Error = (l-y)*$^2$. The error shows the distance between the output and the desired value. The goal of backpropagation is to minimize the sum of all errors for every training sample (Maind

[3]source: https://medium.com/@cosimo.iaia/machine-learning-tensorflow-per-luomo-di-strada-2c71a948b4e3, accessed: 2018-4-1

et al., 2014). This way the NN behaves most desirably. It can be done by adjusting the weights between the neurons. Decreasing the value of the weights in the direction of the gradient leads to the most rapid decrease of the error. This is done with a gradient descent optimizer. With the help of the optimizer the weight vectors are modified for every sample, so that next time, when the same sample is seen, the error is smaller. After training the NN multiple times with the same input, the sum of the error decreases to a minimum value, which means that the model has fitted the data.

The NN described above is the most common and basic NN used for classification tasks. It is called a fully connected feed-forward NN because all neurons between two layers are connected and the input data is moving in a forward direction. There are a lot of other NN architectures, where not all layers are connected but pooled together or where part of the outputs are added to the input of the next sample. In a feed-forward NN, the input samples are assumed to be independent of each other. To predict sequences, the NN needs to get information about past samples, because the samples are not independent. For this task, recurrent NN as described in Mandic (1995) are used. They can be thought of as having a memory that remembers past calculations, or in other words, to every sample a part of the computations from the previous samples is added (Mandic, 1995).

NN are widely utilized in practice and produces satisfying results if there is enough training data. The two significant disadvantages of NN are the long training times and it being a black box model. Even though it may have learned the relationship between the in- and output, we do not know how it learned it, or it can be very complicated to understand.

## 2.4 Decision Trees

The idea of decision trees has been around for many years. A tree consists of nodes and edges which are organized in a hierarchical structure. At the top is the root node, followed by internal (or split) nodes and at the bottom, there are the terminal nodes. Each of these nodes has exactly one incoming and two outgoing edges (for a binary tree). (Criminisi, 2011)
A decision tree is used for making decisions. At the root node, a sample is fed into the tree, and tested. The result of this test decides if the sample is sent to the left or the right child of the root node. There, the sample is subjected to a new test and so on, until it arrives at the leaf node. The leaf node represents the result of the sample. The challenge in creating such a tree is to establish the test functions at each internal node of the tree. The goal of these functions is to

split the samples in a way that samples with the same category always arrive at the same leaf nodes. (Criminisi, 2011)

In general, such decision trees tend to overfit the data, but in recent years they have been revived because it was discovered that ensembles of slightly different trees are less prone to overfit the data and therefore perform much better (Amit et al., 1997, Tin Kam Ho, 1995). A key aspect of ensembles of trees is the fact that its component trees are all randomly different from one another (Criminisi, 2011). This fact helps to decorrelate between the individual tree predictions, improves the generalization, and increases the robustness with respect to noisy data (Criminisi, 2011).
A commonly used example of such an ensemble of trees is the random forest.

## 2.5 Random Forest

A random forest as described by Breiman (2001) is an ensemble method. Ensembles can be thought to be divide and conquer methods, as they consist of groups of weak classifiers, but when they are added together, they form a strong classifier. A random forest consists of multiple decision trees. At each node in a decision tree, a feature is used to split the data into two buckets. With the help of an objective function, the feature that provides the best split is chosen, but the objective function has only a random subsample of features to choose from. (Criminisi, 2011)

When a sample is put into the random forest, it runs down all trees. The average or the weighted average from all trees will be used as the prediction. If the labels are categorical, the majority vote will be used.

The advantages of random forests are that they are fast to train and can deal with missing or unbalanced data. If used for regression problems, it cannot predict values outside the range of the training data. Besides, if the training data is very noisy, it tends to have the overfitting issue (Do et al., 2014).
Random forests can also be used to calculate the feature importance, as described in Strobl et al. (2007). To calculate it, every tree in the forest is traversed, and for every node that splits on feature x, the gini impurity criterion is calculated. For the two descendant nodes, the gini impurity is less than for the parent node. These decreases in gini impurity are weighted by the proportion of samples reaching that node and then added up for every feature. The weighted sum of the decreases in gini impurity gives a good indication of the feature importance for each feature.

However, it should be interpreted with care. It is biased because the feature importance is dependent on the number of categories and the scale of measurement of the feature (Strobl et al., 2007).

## 2.6 Research Gap

The rise of mobile devices and location-based services has led to a significant amount of publicly available mobile data. This fine-grained, multi-dimensional source of data offers new possibilities to tackle established research problems on human mobility (Noulas et al., 2012). There is much research addressing the problem of next place prediction. Different data processing methods have been proposed in combination with different predictors, but most of them use personalized models, which means that for every user a separate model is trained. This is feasible for users with a considerable amount of data, but less suitable for those who only have little training data. Since many models, especially NN, perform better with more training data the data from other users could improve the prediction for users with few data. On the other hand, it could be argued, that the behaviors of different users vary, that the model cannot transfer the learned patterns from one user to predict the behavior of another user. Due to these arguments, an architecture that can model multiple users is chosen, to evaluate how such a model performs. This could be interesting for users with few data because the model could be trained with data from other users and might reach a better performance.

The performance of the model is also highly dependent on the representation of the data. To discretize the labels for each sample, most papers use some sorts of clustering to extract the possible locations for the prediction (Alvarez-Garcia et al., 2010, Burbey et al., 2008), others use a grid to overlay (Sneha et al., 2014, Lin et al., 2012). While both approaches make sense, there does not exist a comparison between the two. The advantage of the clustering approach would be that there are fewer classes that can be predicted, but on the other hand, only places where users have visited before can be predicted.

Last but not least, the choice of the features has a substantial impact on the model performance. While the selection of features is reasonable in the papers presented in section 2.2.2, they are not compared with features used in other papers. For example, it is possible to extract a lot of different temporal features, but it has not yet been described how each temporal feature influences the prediction results. Also, it would be interesting to know how vital a spatial feature such as the distance is in comparison with other temporal features.

# 3 Research Objectives and Overall Methodology

The chosen research questions all serve the same goal. They should help to evaluate the influence of different modeling aspects on the next place prediction problem.
First of all, there are different predictors and different input features that can be used. While other studies predict the next place in one step, in this thesis a hierarchical approach is chosen. Predictions are made on different scales, first on a small scale, then, the scale is increased with every prediction, until the predicted area is small enough. To represent the different scales, hierarchical clustering and rasters with different cell sizes are employed. Since all of the above mentioned factors influence the next place prediction, the influence of each should be thoroughly evaluated.
Based on the research gap identified in section 2.6 and the above mentioned modeling aspects, the following research objectives of this master thesis are defined:

## 3.1 RQ 1: Evaluation of the Model Configuration

Different models can be trained with the same data to compare them. The aim is to find out how the RF and the NN compare to each other and to see if one can handle a specific input better than the other. To see if machine learning is necessary for the task at hand, the two predictors are compared to several baseline predictors. Furthermore, the hierarchical structure of the model will be evaluated to find potential weaknesses.

## 3.2 RQ 2: Data Representation for Input and Output

The data fed to the predictor as input can be represented in different ways. Based on the literature either a raster or a clustering algorithm is used to discretize model labels. If the clustering approach is used, for every cluster, different features are created and used as input. The cluster ID will be used as label. If the grid approach is used, for every cell, different features will be created and the cell ID will be used as label. How do these two approaches compare to each other and what are their advantages and disadvantages?

## 3.3 RQ 3: Input Features

From the available data, different input features can be created. What is the importance of each feature to predict the next place and how does the importance change on different scales?

# 4 Methods

The first section (4.1) of this section explains where the data is from and what it looks like. The next section (4.2) describes how this data was preprocessed and shows two different approaches how places are discretized into a set of areas. Section 4.3 illustrates how the features are extracted from the preprocessed data. With the features ready, in the last section (4.4), it is depicted how these features can be used to predict the next place with an NN and RF, and how they can be used as baselines. At last, the software and the script used in this thesis will be disclosed.

## 4.1 Data

### 4.1.1 GoEco!

The dataset used in this master's thesis was provided by the GoEco! project[4]. It was founded at the ETH and at SUPSI and promotes sustainable mobility behavior by giving feedback about the personal mobility behavior. The GoEco! app helps users to understand how they travel, define their personal goals for change, get personalized suggestions and observe their progress. It aims at investigating if eco-feedback and social interactions (social comparison and peer pressure), automatically provided via information and communication technologies, can be useful in fostering long-term changes in personal mobility behavior and reducing car use (Bucher et al., 2016).

The GPS tracking of the GoEco! app is done with the fitness tracker app called Moves [5]. The user has to permit the GoEco! app to access the personal tracking data, collected in the Moves app. After giving the permission, the data from the Moves app is loaded from the API into the GoEco! app. The data received from Moves is already preprocessed. Outliers, as one would typically have from raw GPS data are already removed. Figure 4.1 shows the structure of the data in the moves app. GPS-fixes are separated into places (stay points) and routes (movement between places). Based on the algorithm developed by Moves, a place is a location where a user stands still for some time. Unfortunately, Moves does not disclose how they extract places. If the user has been at a place in the past, it will always have the same GPS coordinates the next time he or she goes there. A route can have multiple legs if multiple modes of transport (MOT) have been used. The app can distinguish between four MOT namely walking, bicycling, running,

[4]http://goeco-project.ch/index.php/en/, accessed: 2017-12-28
[5]https://moves-app.com/, accessed: 2017-12-28

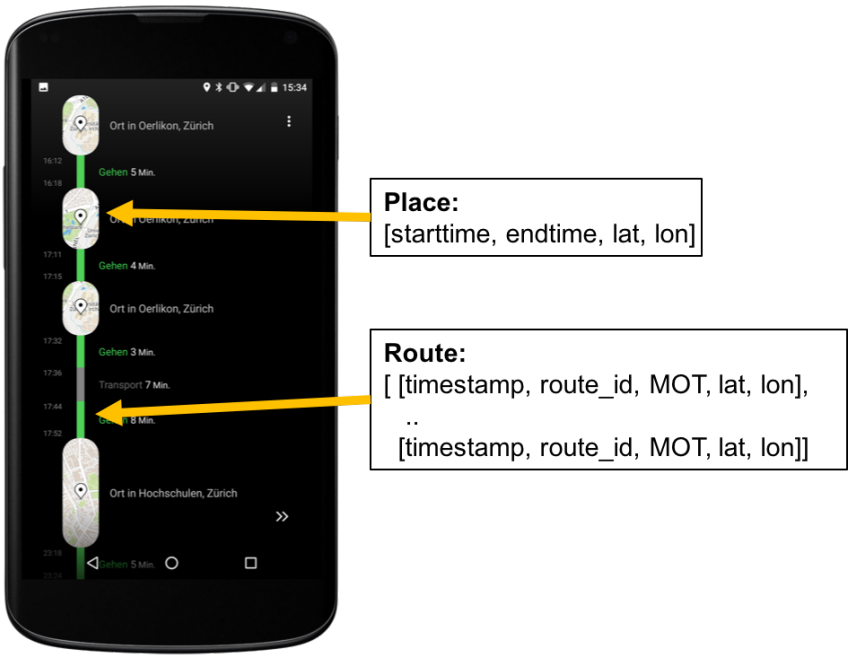



*Figure 4.1: Screenshot of the main view of the Moves app. The GPS data is split into places and routes. A route between two places can have multiple legs if the mode of transport changes. Each route has multiple trackpoints.*

and transport (car, train, etc.).

The routes have a low resolution. On average each route has 17 trackpoints. The sampling rate is 50 seconds on average, but with a standard deviation of 106 seconds, it varies a lot. Typically, there are many more trackpoints in a route if the MOT is walking or bicycling than in transport routes. Therefore, often long routes have fewer trackpoints than short ones. The Moves app does not disclose how its algorithm works, but my guess is that the sampling rate is reduced if the outputs from the accelerometer and the gyroscope are relatively steady to reduce energy consumption. Figure 4.2 shows some examples of routes. In the left picture, there are some very straight routes. These are not GPS errors as one might think, but instead, routes with few trackpoints. Most probably they were recorded in a fast moving train or car where the output of the gyroscope and the accelerometer were relatively steady. The picture on the right shows three example routes that follow the streets relatively well because they contain more trackpoints.

### 4.1.2 Distribution

The places of all users are distributed unevenly over the area of Switzerland. In figure 4.3, many users are visible in the the canton of Zurich and Ticino. This is because the GoEco! project was founded at the ETH and at SUPSI and therefore they specifically targeted people living in the cantons of Zurich and Ticino. In other cantons, only the bigger cities are visible, and in general,

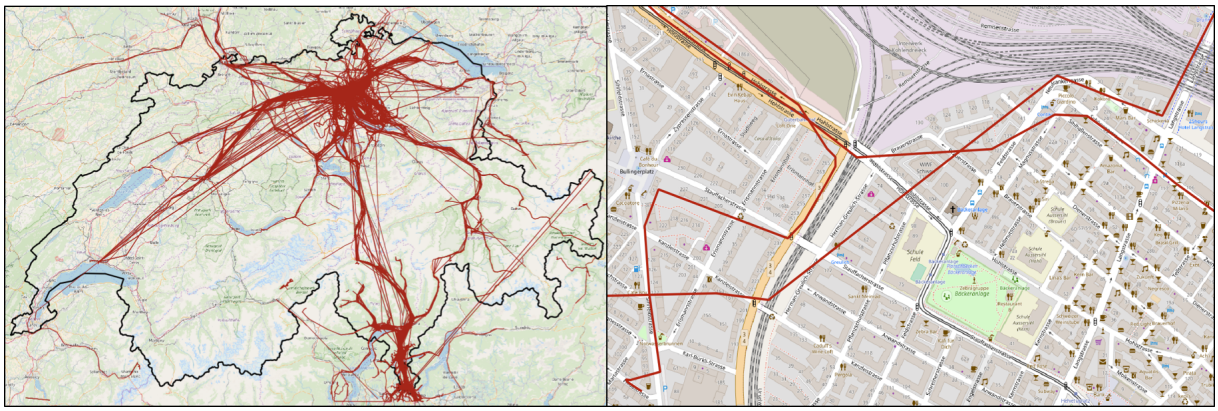


*Figure 4.2: Examples of routes. On the left, all routes in Switzerland are plotted, while on the right only three different routes are shown, but on a larger scale*

there are fewer users in the French-speaking part than in the rest of Switzerland.

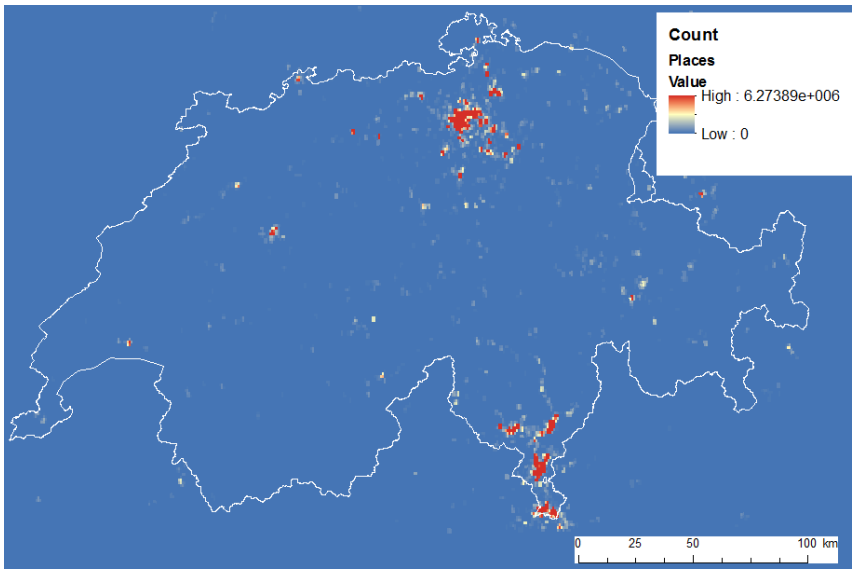



*Figure 4.3: Point density map of all places within the bounding box of Switzerland. The higher this value is, the more points were encountered in the radius.*

Figure 4.4 shows the distribution of all trackpoints in the area of Switzerland. In the canton of Zurich there are nearly no paths observable because there are so many trackpoints. The same thing applies to the canton of Tessin, but contrary to Zurich there are some really distinctive paths, most probably because movement is restricted by mountains and because there are fewer users. In general, most cities appear as yellow dots and the main mobility axes are visible.

### 4.1.3 Data Selection

The dataset contains records from 712 different users. Most of the records are in Switzerland because the GoEco! project is based in Switzerland. Since the data outside of Switzerland is very sparse, only points within the bounding box of Switzerland are used. Within the bounding

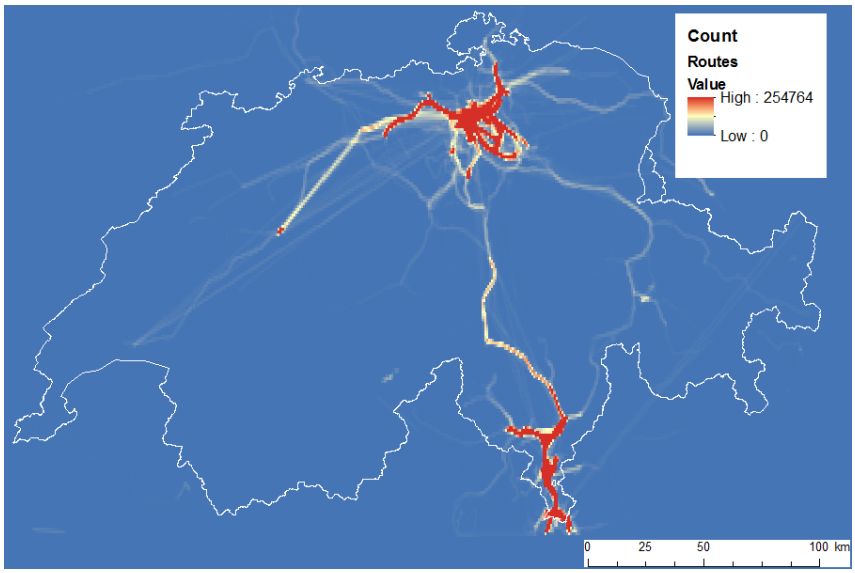



*Figure 4.4: Line density map of all routes within the bounding box of Switzerland.*

box there are 544 users.

To predict a user's next location, there are several factors that need to be considered. First of all, a model can only predict places where a user has already been. Second, users with few data are tough to predict because it is difficult or impossible to extract their habits. Third, because of the data structure there are some places in the dataset, where a user has been for a brief period, maybe because he or she talked to someone or because the user lost the orientation. Such places are also very challenging to predict because they often happen at random. To simplify the prediction, all users with less than 10,000 trackpoints were removed. From the remaining 228 users, 41 were selected randomly to increase preprocessing and training speed. The data of these users was subjected to the following procedure:

1. Places with less than six visits are disposed and the trackpoints leading to these places as well.

2. Places, where a user stayed for less than 15 minutes are dropped, as well as the trackpoints leading to these places.

After the selection process, the datasets have some distinct differences. Figure 4.5 shows the cumulative distribution of the number of trackpoints and places per user. Looking at the 41 users it is evident that this dataset contains fewer users with few places.
This is because only places with 6 or more visits are used, which means that users who used the app briefly will be ignored. Additionally, only users that have more than 10,000 trackpoints are included, which also increases the number of places per user. Looking at the trackpoints per user, the selection also includes fewer users with little data for the same reasons as mentioned

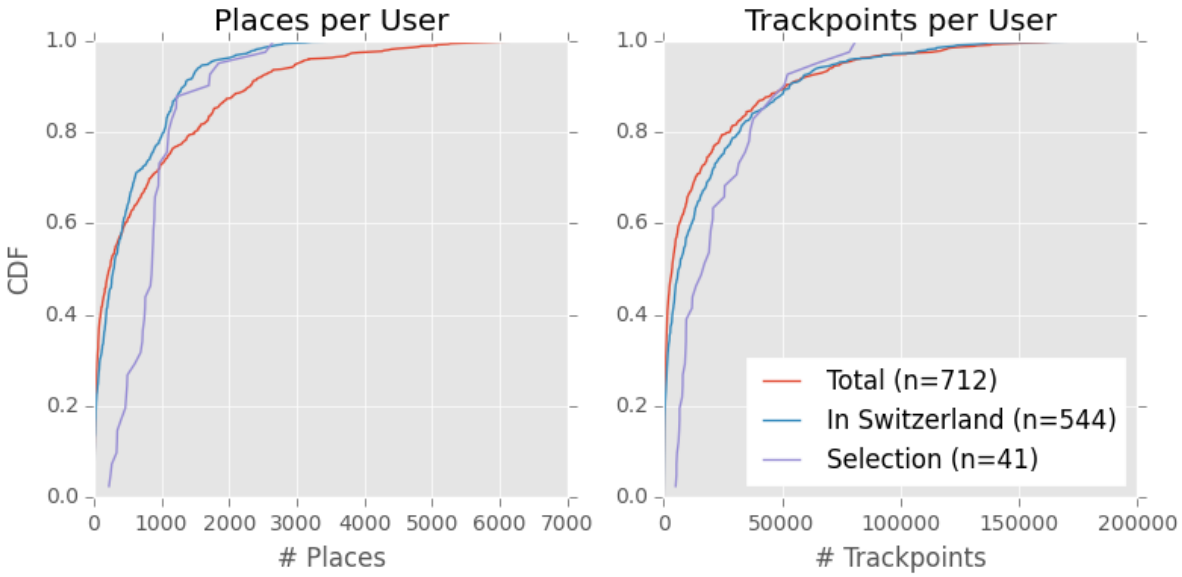



*Figure 4.5: Cumulative distribution of the number of places / trackspoints per user. In the selection, only places and trackpoints that fulfil the requirements mentioned above are included.*

above. The minimum amount of trackpoints per user lies around 10,000 because in a first step users with more than 10,000 trackpoints were selected and then all trackpoints that lead to a place where the user has been less than six times or less than 15 minutes are removed.

| | Total | in Switzerland | 41 users |
|---|---|---|---|
| routes | 657389 | 588,556 | 48795 |
| route trackpoints | 11,428,382 | 9,860,235 | 964,026 |
| places | 545,165 | 481,822 | 37,182 |
| users | 712 | 544 | 41 |
| places per user | 766 | 886 | 906 |
| routes per user | 944 | 1100 | 1190 |
| trackpoints per user | 16,051 | 18,125 | 23,513 |
| trackpoints per route | 17 | 17 | 20 |
| route sampling rate | 53 s / 541 m | 53 s / 392 m | 44 s / 331m |
| average route distance | 8.9 km | 6.1 km | 6.2 km |

*Table 2: Data description*

Typically most datasets contain many users with very few data and few with plenty data. Since it is very tough to extract patterns for users with few data, the figure clearly shows, that there are fewer users in the selection with few data, which means that the model should perform better than with the other two data sets. Therefore, the requirements mentioned above are useful.

Another way of looking at the data is to visualize the number of days in which each user collected data. Figure 4.6 shows that in the random selection there is no user with less than 43 days. This is very important since temporal patterns should be extracted from the data, which

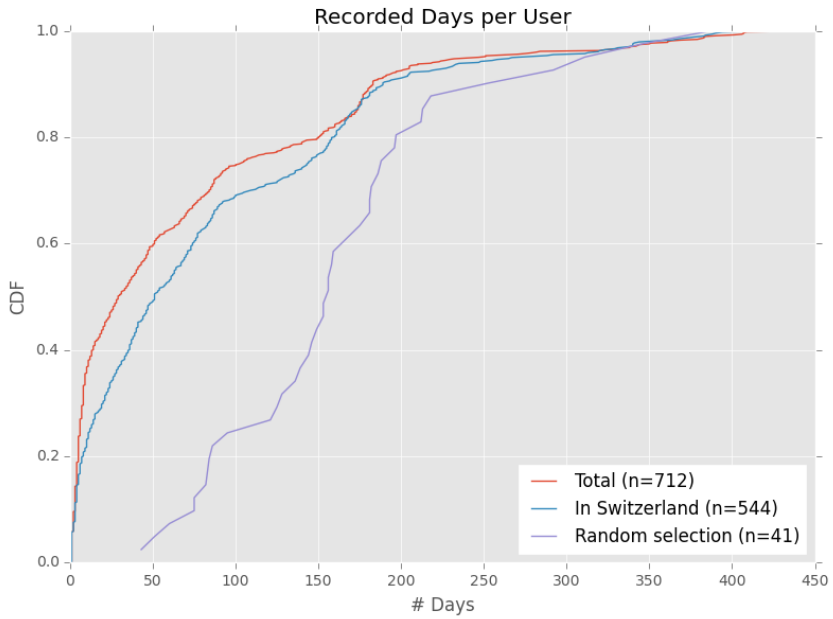



*Figure 4.6: Cumulative distribution of the days with recording for every user.*

is very difficult if the user collected data for only a few days. On average, a user has collected data for 162 days and the median is at 156. Therefore, it should be possible to find meaningful patterns.

In general, it can be said that the data provided by the GoEco! project has a skewed distribution. Many users collected data only for a short amount of time. Therefore, they only have few trackpoints and places. In the random selection of 41 users, the distribution is less skewed, because users with few data were not taken into account in the selection. This and the other two requirements should make it possible to extract a users movement patterns, which ensures that the input data is useful.

## 4.2 Place Discretization

Since it is difficult to predict x and y coordinates, places are segmented into regions with an ID. To group all places, two different methods were used. In the first approach, all the places are clustered and each cluster is assigned a unique ID. In the second approach a raster is used to aggregate all places into cells and assign them a unique cell ID. Since the places are diffused over the entire area of Switzerland, the segmentation is done hierarchically. On the top level of the hierarchy, a wide-meshed raster and a small number of clusters is used. Since the area of these cells or clusters is large, the prediction is rather simple, because the users mostly do not change the cluster/cell. Next, the clusters are clustered again and the cells are subdivided into

smaller cells. This process is then repeated to get a predicted area that is small enough. The challenge is to find a suitable amount of clusters and cells. If a high number of clusters/cells is chosen the area of each cluster/cell will be small and fewer levels of the hierarchy are needed. On the other hand, the prediction will be more difficult since there are more clusters and cells that possibly can be predicted. For example, if on every level, the data is split into 4 clusters/cells, the prediction will be quite simple because there are only four areas that can be predicted. So, if chosen randomly the accuracy would be at 25%. On the other hand, this would require many levels until the clusters/cells have a reasonably small area that can be used as the final prediction. On the first level of the hierarchy, 100 cells and clusters were chosen. For the raster, the number of cells stays the same in the second and the third level. In clustering approach, 30 clusters were chosen for the second and the third level.

### 4.2.1 Clustered Input

Clustering the places of all users creates areas, whose IDs can be used as the output of the NN. Since the places stem from multiple users, the clusters do not have a real meaning, hence should not be interpreted. The clusters just describe areas where some people have been, and their borders are defined by areas where no user has been so far.

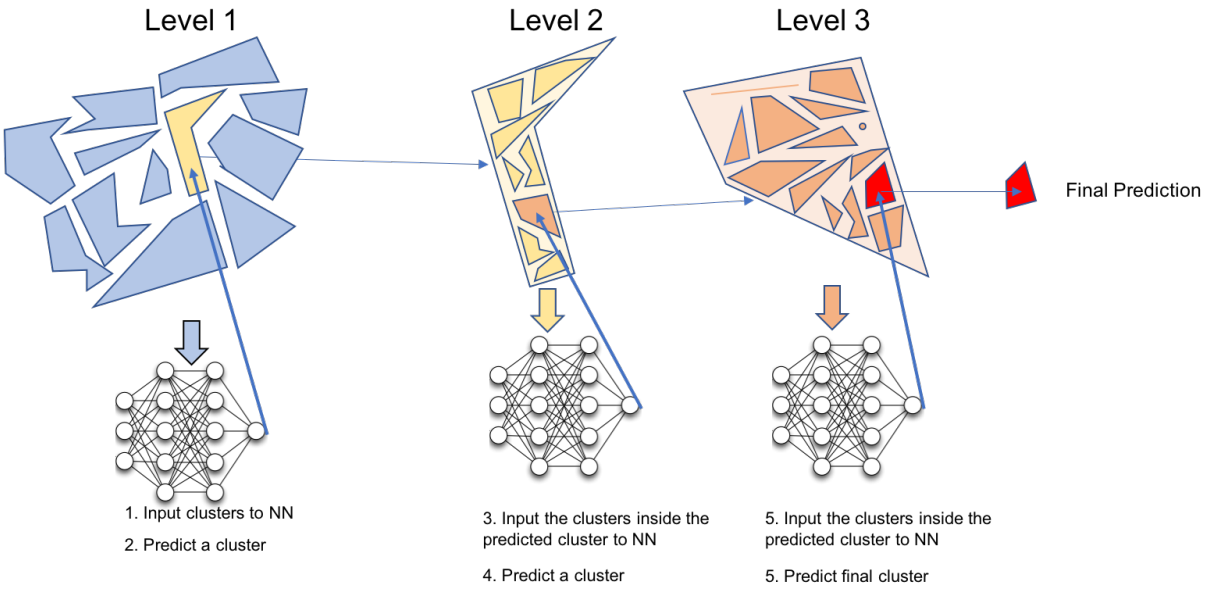



*Figure 4.7: The process of the hierarchical prediction with clusters. Different layers of clusters at different scales.*

Figure 4.7 shows what the prediction with the clustered input looks like for each level. In a first step, all places are clustered into 100 clusters. All clusters lying within the cluster predicted from

the NN in level one will be used as possible predictions for the NN in level two. Then again, all clusters lying within the predicted cluster from level two will be used as possible predictions for the NN in level three. The prediction by the NN in level three will be the final prediction. Figure 4.8 shows an instance containing real data.

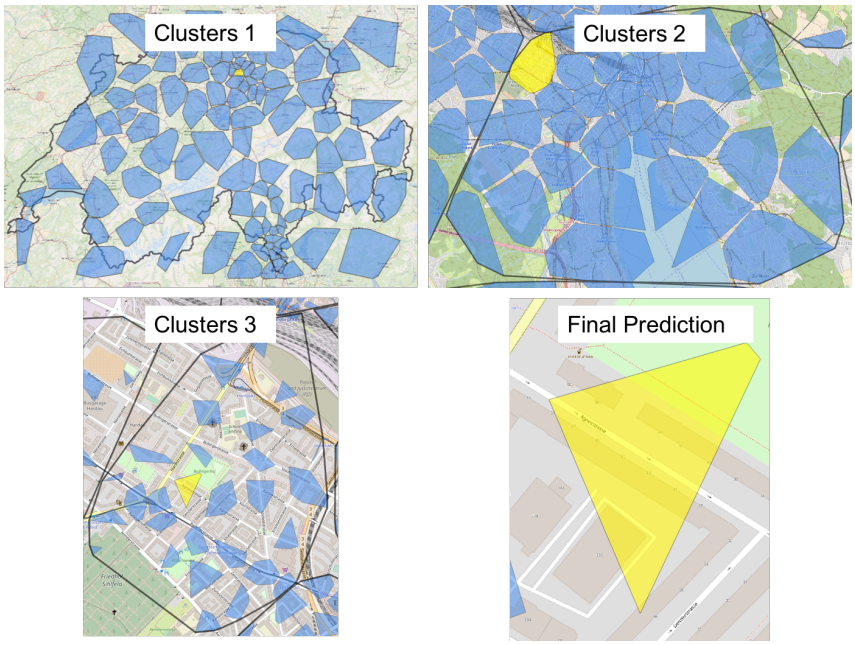



*Figure 4.8: Different layers of clusters. The areas show the convex hull of all places within the same cluster.*

In the selection of a clustering algorithm, the most crucial issue is that the number of clusters can be set manually. This is because the size of the input vector always needs to be maintained. This can only be accomplished if the number of clusters does not exceed an absolute maximum. If the number of clusters is below the maximum, the missing cluster data can be filled with negative numbers. The second criterion is, that the algorithm should be fast. K-means is a good choice because it satisfies both criteria.

K-means is an iterative algorithm. At the start, a number of cluster centers can be set. These centers are randomly distributed in space. All data points, in this case places, are assigned to the nearest cluster center. Next, the centroid for every cluster is calculated to update the cluster centers. This process is repeated until the cluster centers no longer move or some stopping condition is met. A stopping condition can be that the sum of the cluster center offsets between two iterations are below a threshold, i.e. the cluster centers barely move or that the maximum number of iterations is surpassed. In this thesis, the k-means implementation from the scikit-learn[6] library was used.

In level one, the places are clustered into 100 clusters. Each of these 100 clusters are clustered

[6]https://scikit-learn.org, accessed: 2017-12-28

again into 30 sub-clusters in level two and once again in level three. For clusters that contain less than 30 places, each place is treated as a cluster. This means that a cluster can be a line or a single point as well because it may consist of only one or two points. For this reason, some clusters in the second level cannot be clustered again since they contain only one place. Hence the prediction in level two can be the final prediction in some cases. If every cluster in level one were split into 30 sub-clusters, there would be 30*100 = 3000 clusters in level two, and if all these were split again into 30 sub-clusters, there would be 3000*30 = 90'000 clusters in level three. Since there are many clusters in level one and two that contain less than 30 points, the number of clusters in level two and three is not nearly as high (table 3).

| | Level 1 | Level 2 | Level 3 |
|---|---|---|---|
| Average area | 374.5 $km^2$ | 506'906.3 $m^2$ | 424.1 $m^2$ |
| Standard deviation area | 299.7 $km^2$ | 916'285.2 $m^2$ | 1650 $m^2$ |
| Nr. of clusters | 100 | 2980 | 29'270 |

*Table 3: Average and standard deviation of the cluster area (convex hull) and the number of clusters in each level*

### 4.2.2 Raster Input

Segmenting the places of all users into raster cells creates areas whose IDs can be used as the output of the NN. The size of the cells was chosen to be divisible by ten because on every step down in the hierarchy because each cell will be divided by a new ten by ten raster. Additionally, since the first raster has to contain the entire bounding box of Switzerland, the width of the first raster needs to be bigger than the width of the bounding box of Switzerland. Since Switzerland is approximately 350 km wide, a 400 km raster and therefore a 40 km cell size was chosen for the first level. Table 4 shows the relevant information for every cell.

| | Level 1 | Level 2 | Level 3 |
|---|---|---|---|
| Cell area | 1600 $km^2$ | 16 $km^2$ | 0.16 $km^2$ |
| Cell width | 40 km | 4 km | 0.4 km |
| Nr. of cells | 100 | 10'000 | 1'000'000 |

*Table 4: Cell area and the number of cells in each level*

Figure 4.9 shows what the prediction with the raster input looks like for each level. In a first step, all places are segmented into 100 cells. The cells from level one are segmented into a finer raster for level two and then again for level three. All cells lying within the cell predicted from the NN in level one will be used as possible predictions for the NN in level two. Then again, all

cells lying within the predicted cell from level two will be used as possible predictions for the NN in level three.

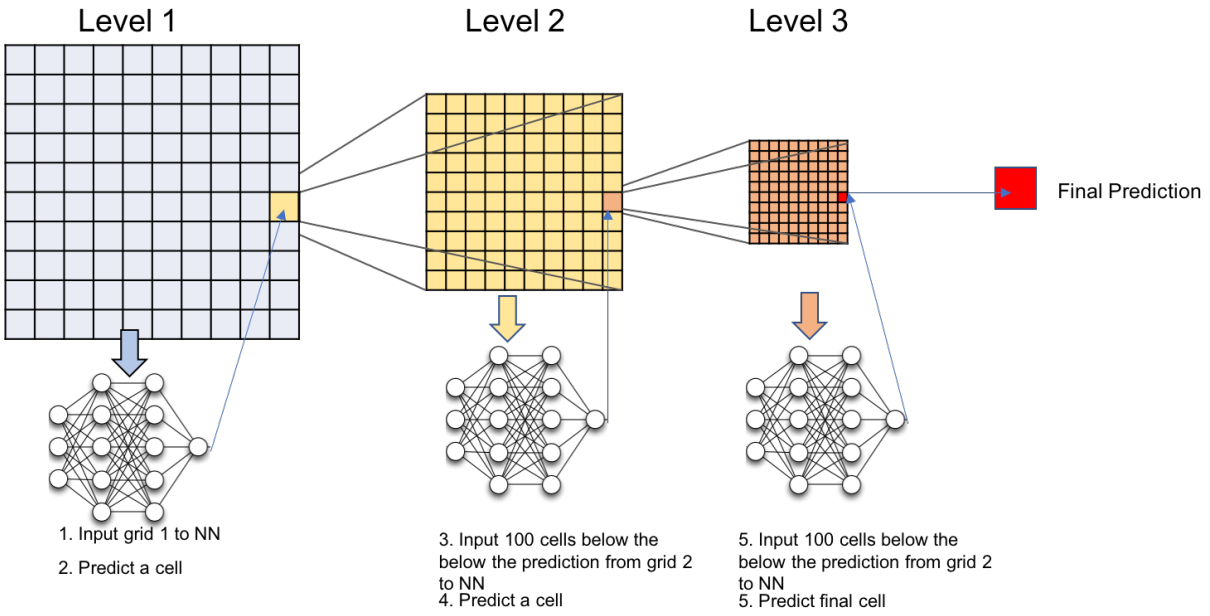



*Figure 4.9: Different layers of rasters*

The prediction by the NN in level three will be the final prediction. Figure 4.10 shows an example of real data. The two top and bottom rows of the raster of level one are not contained in the bounding box of Switzerland. Therefore, they will never have any points inside. This is not an issue because it just makes the prediction easier since there are only 60 instead of 100 real choices.

### 4.2.3 Centroid Prediction

After predicting the cluster or cell on the third level of the hierarchy, a further step is made to predict a coordinate. To predict a point and not an area, the centroid of all places of this user within the predicted spatial unit is computed. This centroid is used as the final prediction. It can be used to measure the distance between the prediction and the actual place where the user went. To compare different predictors, the average distance deviation between the predictions and the labels can be used in addition to the accuracy.

## 4.3 Cluster / Raster Features

Each cell or cluster can be described with a set of features. It is known where the user has been in the past. Thus, it can be counted how often he or she has visited a specific spatial entity. These counts can be used to create further features. As proposed by Wang et al. (2012) or Gomes et al. (2013), the time stamp can be used to create features that describe periodicity. For instance,

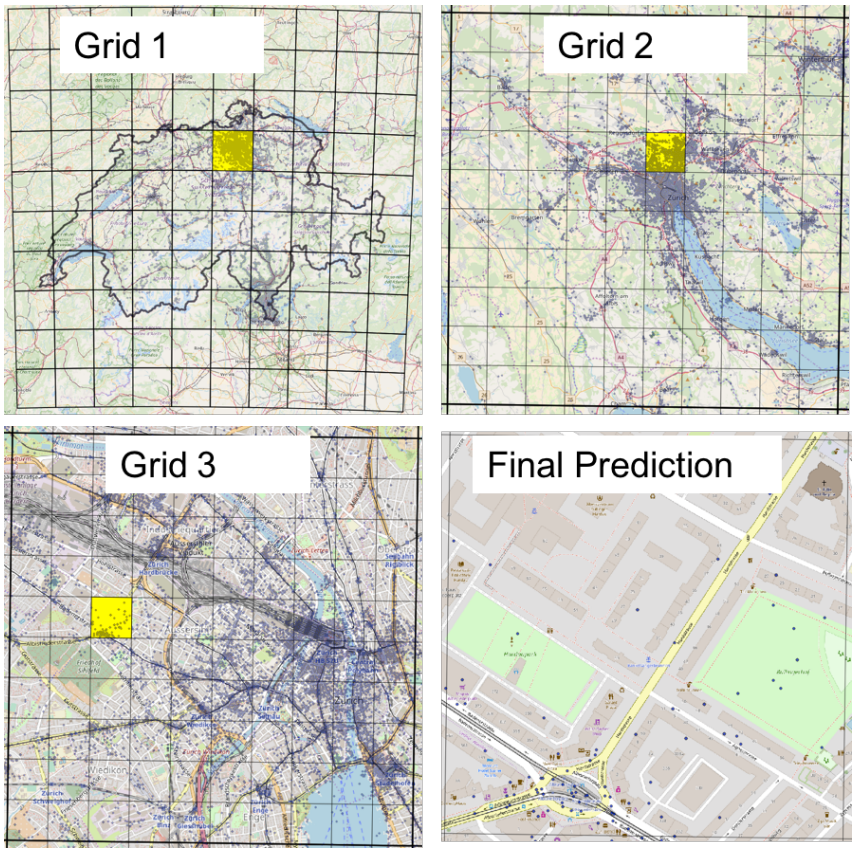



*Figure 4.10: Different layers of rasters*

how often the user has been at each cluster/cell on a weekend or during the week, or how often he or she has been there on a Monday.

In addition to the temporal features, spatial features, such as the distance or the azimuth between the user and the cells/clusters can be used to create spatial features as in Noulas et al. (2012). Based on these geometrical and temporal properties, the following seven input features were chosen.

### 4.3.1 Three Hour Window (*h3*)

This feature consists of a count in every cell/cluster where the user has been in a three hour window. The current hour is taken and all places where the user has been within plus or minus one hour are counted for each spatial entity. For example, if it is 3 p.m., for every cell or cluster it will be counted how often the user has been there between 2 and 5 p.m. This feature should help to describe patterns that a user typically does at a specific time of day. As an example, this could be drinking coffee at the favorite bar in the morning or buying groceries at the nearby shop after work.

### 4.3.2 Day of the Week (*dow*)

This feature consists of counts of all places where the user has been on the same day of the week. This feature should help to describe patterns that a user does on the same days of the week, such as going to a sports club or attending an evening course.

### 4.3.3 Week or Weekend (*week*)

This feature consists of counts of all places where the user has been on weekdays or weekends. If the current date lies on the weekend, all places visited during weekends are used to get the counts for each spatial entity, otherwise, all places visited during weekdays. Since most users work on weekdays and enjoy their leisure time on weekends, this feature helps to describe work and leisure time patterns. As an example, this could be going to work on weekdays and going out on Saturday night.

### 4.3.4 Total Count (*tot*)

This feature just describes the overall place count in each cluster or cell. This is beneficial since the prediction is already pretty good if always the spatial entity with the highest count is predicted. It is especially useful to have the total count as a backup in cases where the other temporal features described above are not helping.

### 4.3.5 Distance (*dist*)

This feature describes the distances between the user's current position to every cluster/cell. The distance is computed from the current position to the nearest point of the spatial entity. This feature helps the predictor to see which spatial entities are close to the current position. To get the distances, the database is queried using SQL and the extension PostGIS.Either the minimum distance from the current position to every cell is calculated, or the nearest distance from the current position to the convex hull of each cluster is calculated for the raster and cluster input respectively.

### 4.3.6 Azimuth (*azim*)

This feature describes the angle between the direction in which the spatial entity lies relative to the current movement direction of the user. The direction of the cell/cluster is calculated as azimuth between the centroid and the current position of the user. The movement direction is computed by averaging the directions between the trackpoints of the current route. This feature helps the predictor to see which spatial entity lies in the movement direction. Figure 4.11 shows how the direction is calculated.

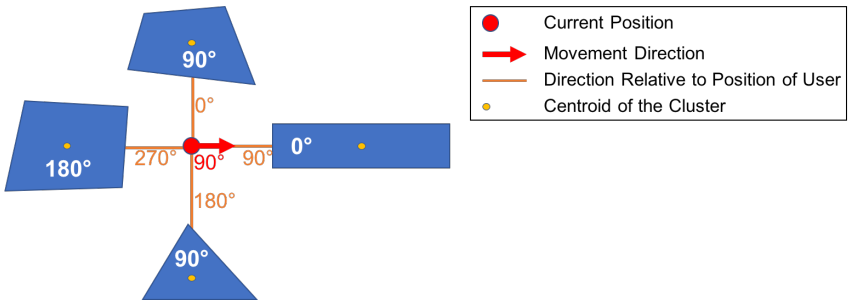



*Figure 4.11: Azimuth calculation. The result is always the absolute value of the movement direction minus the direction of the spatial entity. If the result is bigger than 180, 180 will be subtracted, since 180 is the exact opposite direction.*

The azimuth between the user's position and the centroid of the spatial entities is queried first and then the absolute value of the difference between these values and the average movement direction is calculated. Since the maximum angle between the movement direction and the spatial entity can only be 180, 180 will be subtracted if the result is higher than 180. Therefore the real direction of the spatial entity is not considered, but only the direction in relation to the movement direction.

### 4.3.7 Similar Routes (*act*)

Since the other features are derived from spatial units, the last feature is derived from the routes. Most people always take the same route whenever they go to a particular place, so if a user is moving somewhere, one can test if he or she has been there before and where he or she went last time. Sometimes, a user has passed by the same place, but he or she was moving in the opposite direction. For example to go from home to work and going back home after work. Therefore it is helpful to compare the current movement direction to the average direction of the route and only consider routes with a similar direction. Figure 4.12 shows how the similar route feature is calculated.

In a first step, the trackpoints from every user are converted to lines, where each line represents a route. Next, a buffer with a radius of 100 meters is drawn around the current position of the user. All routes that stem from the same user and overlap the buffer are retrieved. From this selection, all routes with an average direction that deviates more than 90 degrees from the current movement direction, are removed. This means that only nearby routes with a similar direction are left. Since it is known in which spatial unit these routes end up, the ID of the unit can be retrieved. The count for each distinct cluster / cell ID is used for the prediction.

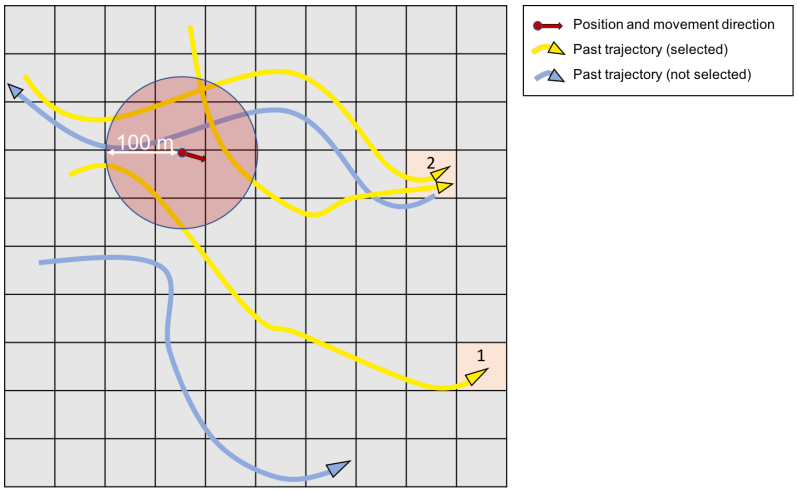



*Figure 4.12: Similar routes calculation. Only past routes within a radius of 100 meters, whose average movement direction deviates less 90 degrees from the movement direction of the user are considered.*

## 4.4 Next Place Prediction

### 4.4.1 Input Vector

The seven features described in section 4.3 are used to describe one cluster or one cell. Each spatial unit is described with a feature vector of seven numbers and the feature vectors of every spatial unit form one input vector. For the rastered input on every level the input has 700 numbers because each raster contains 100 cells. For the clustered input the first level input contains 700 numbers as well, but on the second and third level, there are only 210 numbers per input because there are only 30 clusters in these levels.

### 4.4.2 Dimensionality Reduction

There is a lot of redundant information in the input vector. For instance, if a user visits a particular place only on Sundays, the *dow* feature and the *week* feature are the same. In general, the temporal features are correlated. Since dimensionality reduction techniques summarize correlations in a set of variables into a smaller set of variables, two dimensionality reduction techniques are tested. This should simplify the input as well as increase the training speed, since less data needs to be processed.

The two most common dimensionality reduction techniques are the linear discriminant analysis (LDA) and the principal component analysis (PCA) (Martinez et al., 2001). The PCA does not need any class labels, it just tries to find the directions, which are also called components, in which the dataset has the highest variance. The LDA needs class labels because it tries to find the directions, called linear discriminants, that best discriminate between classes.(Martinez et al., 2001)

These techniques are employed before feeding the data to the classifier to make the input data denser. After training and testing, the results can be compared to find out which technique works better or whether they improve the result at all.

### 4.4.3 Neural Network

The library tflearn[7], which is a higher level API for tensorflow[8] is used for the neural network. As network architecture, a simple feed-forward architecture was chosen. For a feed-forward neural network, three main parameters must be set. The first is is the number of hidden layers. If the data is linearly separable, then no hidden layer is needed, but since this is not the case, at least one hidden layer should be used. To find the optimum number of hidden layers, multiple NN are trained with different amounts of hidden layers. The best accuracy is reached with two hidden layers. The second parameter is the number of neurons per layer. To find the number of neurons, the rule of thumb proposed by Blum (1992) was used, which is summarized in the following formula:

$$\frac{N_\mathrm{i} + N_\mathrm{o}}{2} = N_\mathrm{n}$$

$N_i$ = *Number of input neurons*
$N_o$ = *Number of output neurons*
$N_n$ = *Number of neurons in hidden layer*

The third parameter is the number of epochs. In one epoch the NN is fed with all training data. The number of epochs describes how often a sample is shown to the neural network. If the number is too high, the NN starts to overfit and if it is too low, the NN has not learned everything it could and therefore may still improve. Since backpropagation is an optimization problem which seeks to minimize its loss function, we can we can look at the loss to find the optimal number of epochs. If the loss does not decrease any more, the NN stopped learning anything. Fortunately, there is an extension for tensorflow called tensorboard, which shows graphs of the loss and the accuracy of the NN while training. According to these graphs, the accuracy does not increase anymore and the loss does not decrease anymore after the sixth epoch. Therefore, six epochs are chosen.

[7] http://tflearn.org/, accessed: 2017-12-28
[8] https://tensorflow.org, accessed: 2017-12-28

### 4.4.4 Random Forest

For the random forest, the algorithm from scikit-learn[9] was used. The only parameter that was varied was the number of estimators/trees. To find the optimum for the number of estimators, several random forests were trained with different numbers of estimators. With the given data the optimal number of estimators is around 200. Therefore the random forest was trained with 200 decision trees.

### 4.4.5 Baselines

Our predictor is compared with some baselines which are very simple predictors. If a baseline performs better than a more complicated predictor such as the NN or the RF, it shows that these predictors do not learn much from the input. Since the features contain a number for every spatial unit, the next cell/cluster can be predicted directly. The temporal features contain counts for every spatial unit and the distance features contain the distance to every spatial unit. Therefore the spatial unit with the highest or the lowest number is chosen as prediction. The *azim* feature is not used as a baseline because typically multiple cells or clusters lie in the current movement direction, which makes a prediction difficult. Table 5 gives a summary of all baselines.

| Name | Feature | Prediction | Method |
|---|---|---|---|
| $B_{dist}$ | *dist* | Nearest spatial unit | Lowest number |
| $B_{act}$ | *act* | Most visited spatial unit at the end of routes from the past, which resemble the current route. | Highest count |
| $B_{tot}$ | *tot* | Most visited spatial unit | Highest count |
| $B_{h3}$ | *h3* | Most visited spatial unit in a three hour window | Highest count |
| $B_{dow}$ | *dow* | Most visited spatial unit on day x of the week | Highest count |
| $B_{week}$ | *week* | Most visited spatial unit on weekdays / weekends | Highest count |

*Table 5: Summary of the baselines*

### 4.4.6 Workflow

Figure 4.13 shows the first preprocessing step, which is mostly done with SQL. The data is loaded from the Moves API. It returns a file for every user with all places and trackpoints. In a first step all points outside the bounding box of Switzerland are removed. Since finding patterns is complicated if a user has only little data, all users having less than 10'000 trackpoints are

[9] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, accessed: 2017-12-28

deleted. From the remaining users, 41 are chosen randomly. The data from the remaining users is stored in a table for places and in a table for trackpoints. For every trackpoint, the average azimuth of the preceding trackpoints in the route is calculated. To know where the route will end up, the place ID of the next place is added to every trackpoint. For every place, the duration of the stay and the number of visits is calculated. To calculate the number of visits, a buffer with a radius of 20 meters is drawn around every place and the places of the same user within the buffer are counted. If the number of visits is less than six or the duration of the stay is less than 15 minutes, the places are deleted. After deleting these places, all trackpoints that lead to deleted places are deleted as well.

The places are now clustered and rasterized as described in section 4.2, and the result for every level of the hierarchy is added to the places table.

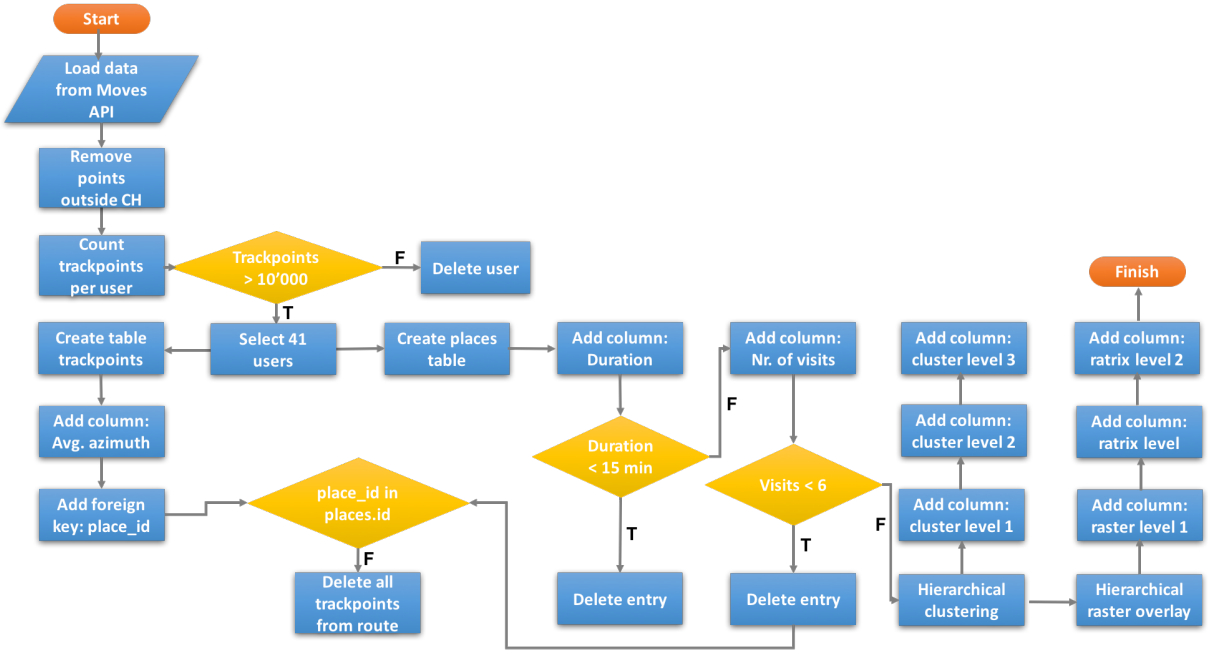



*Figure 4.13: Flow Chart of the preprocessing*

In a second step, the data in the two tables is used to create input features (figure 4.14). For every user, the trackpoints are loaded and for every trackpoint the seven features are computed. These features are stored separately for every user as a serialized python objects (pickle). The labels / output is computed by getting the cluster / cell ID where the route is ending up.

The training and testing is explained in figure 4.15. After creating serialized python objects (pickles) of all features, they are loaded and normalized. The features are merged into a feature vector and then appended to an array. The pickle containing the labels is loaded as well and the labels are converted into a one-hot array. Having loaded the input and the labels of all users,

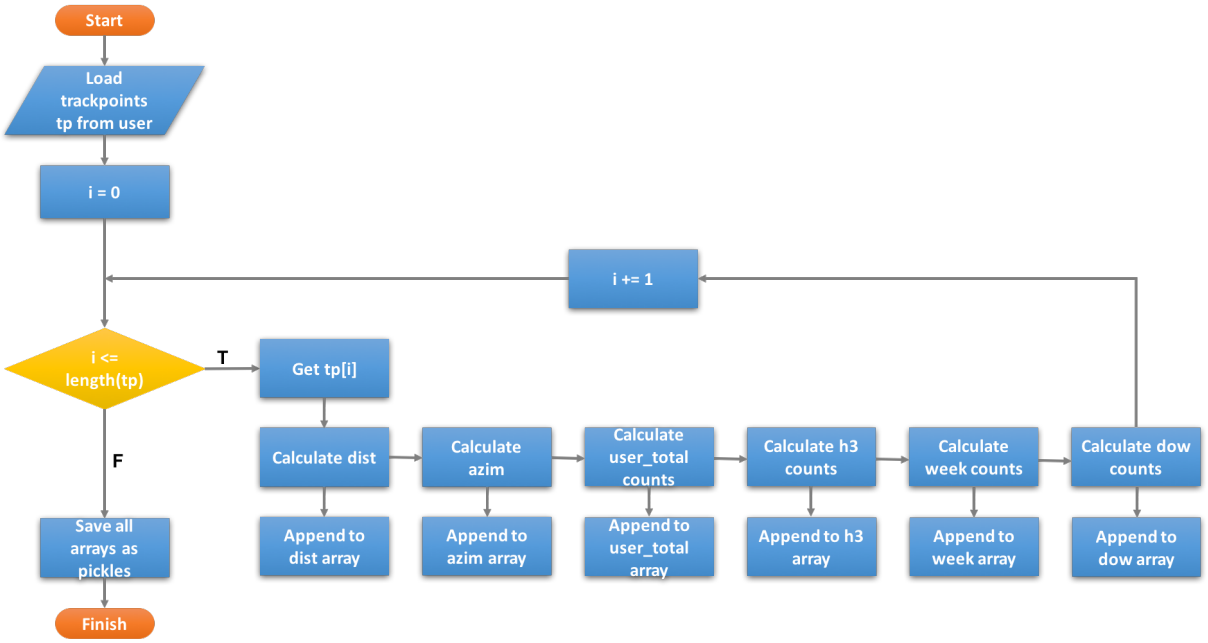



*Figure 4.14: Flow Chart of the feature calculation. This process is done for every user three times, once for every level in the hierarchy. Additionally, everything is done once for the cluster and once for the raster input*

they are split into training and testing data. The newest 10% of every user go to the test set and the other 90% into the training set. In some model runs the test and training data is also projected into a subspace with principal component analysis or linear discriminant analysis, to reduce the number of inputs. After that, the random forest and the neural network are trained. For the evaluation of the results, first the baselines are computed first, then the test data is used to get the accuracy for the NN. To have another measure of the performance the distances between the ground truth and the centroid of the NN prediction are calculated. Next, the test data is used to calculate the RF accuracy, and lastly, the results are saved.

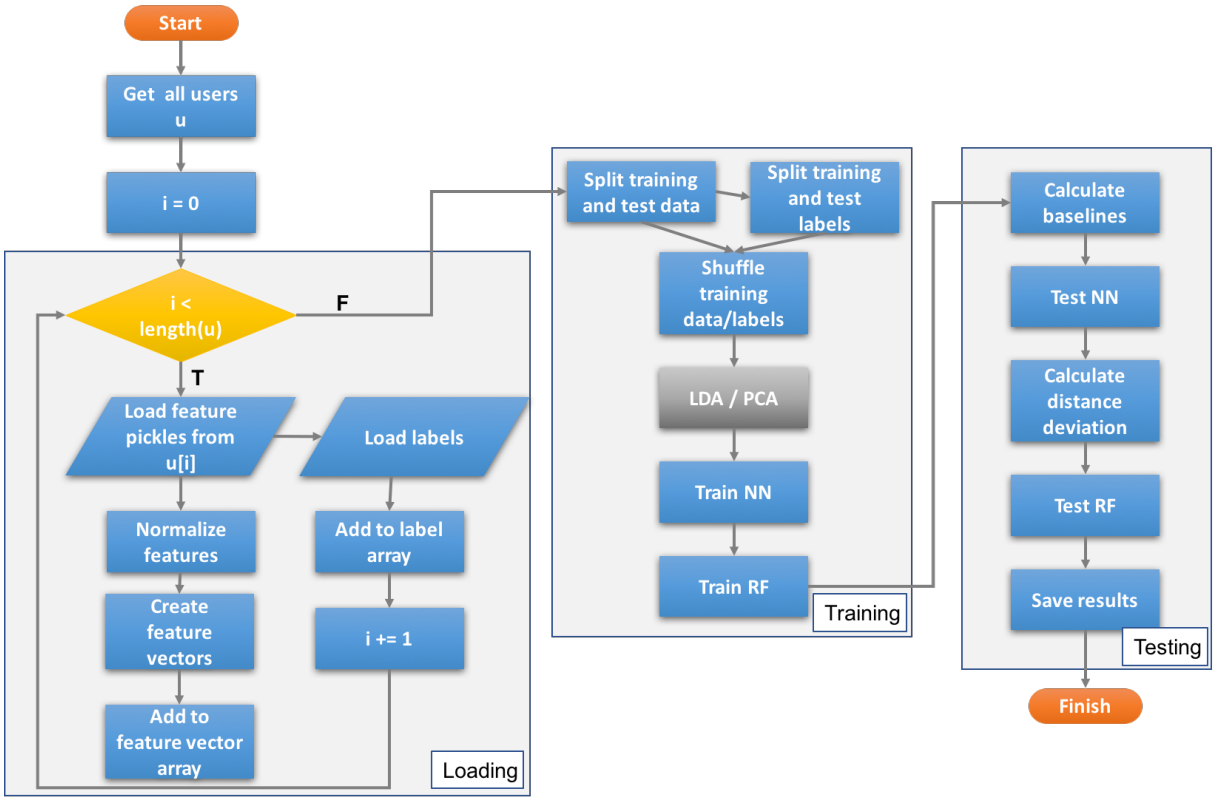



*Figure 4.15: Flow Chart of the loading, training and prediction. The features are loaded and merged into a feature vector, which is used for training and testing.*

## 4.5 Software and Scripting

Most of the processing steps were scripted with the programming language Python[10]. Since there are a lot of third-party packages available for Python, additional software such as database management systems (PostgreSQL[11]) and machine learning packages (tensorflow[12], scikit-learn[13]) could be incorporated in the scripts. Python was not only used for the data preprocessing and the modeling but also to create figures and graphs (matplotlib[14]).

The data stored in the SQL database could be queried in python scripts, manipulated and then stored back into the database. A handyl extension for PostgreSQL was the spatial database extender Postgis[15], which allows location queries to be run in SQL. It enabled the querying of spatial relations and distances between objects. Thanks to Postgis the data could be visualized in QGIS[16], a free, open source geographic information system. QGIS was very helpful to display the data on a map, get an overview of the data and to look at the predictions.

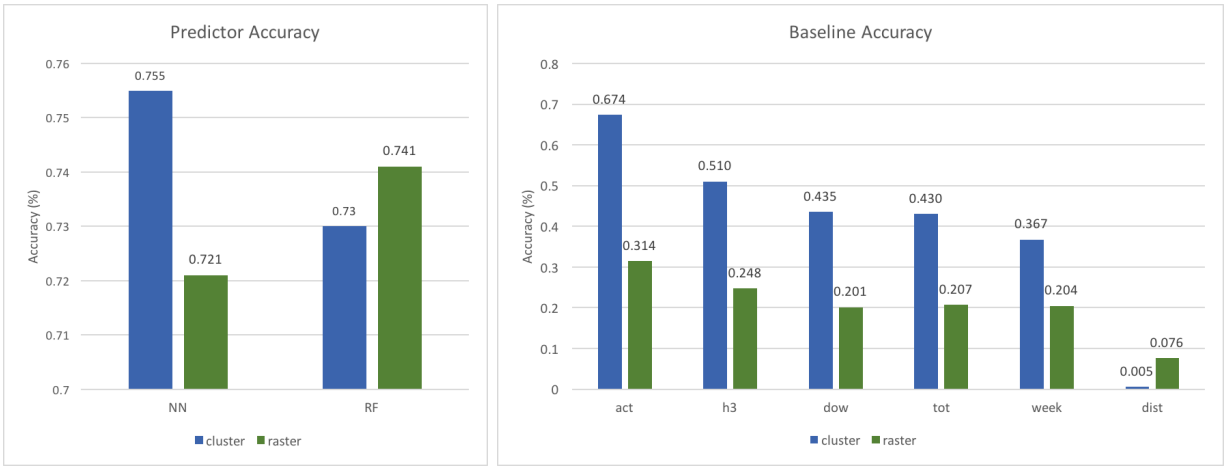
[10] http://python.org, accessed: 2017-12-28
[11] https://www.postgresql.org, accessed: 2017-12-28
[12] https://tensorflow.org, accessed: 2017-12-28
[13] https://scikit-learn.org, accessed: 2017-12-28
[14] https://matplotlib.org/, accessed: 2017-12-28
[15] http://postgis.net/, accessed: 2017-12-28
[16] https://qgis.org/en/site/, accessed: 2017-12-28

# 5 Evaluation and Results

First, it will be shown how the predictors and the baselines performed in terms of accuracy (section 5.1). In a second step, the distance between the predicted and the actual place will be evaluated (5.2). Instead of just looking at the predictor accuracy, one can also inspect the accuracies of the predictor per user and compare them to the results of predictors that were trained for each user separately (5.3). Lastly, the feature importance given by the RF will be evaluated (5.4).

## 5.1 Predictor and Baseline Accuracy

The predictor accuracy describes the ratio between the number of correctly predicted and the total number of samples. In general, the predictors and the best baseline performed reasonably well as shown in figure 5.1. With the clustered input, the NN predicted the next location in 75.5% of the cases. It performs best, even though the RF comes in close as second best with an accuracy of 74% using the rastered input. The RF with the clustered input reaches an accuracy of 73% and the NN with the rastered input 72%.




(a) Predictors

(b) Baselines

*Figure 5.1: Accuracy comparison*

Looking at the baselines in figure 5.1(b), it can be observed that the baseline $B_{act}$ clearly outperforms the others. This is true for the clustered and the rastered input. With an accuracy of 67.47% for the clustered input, it outperforms the others by more than 16% and it is only 5% worse than the worst performing prediction algorithm. For the rastered input, $B_{act}$ does not perform nearly as well, and reaches only 31.4%.

While the $B_{dow}$, the $B_{tot}$ and the $B_{week}$ perform fairly similar with the raster input, the $B_{week}$ is outperformed by the other two when fed with the clustered input. Finally, the $B_{dist}$ clearly

performs worst with only 7% using the rastered input and 0.5% using the clustered one.

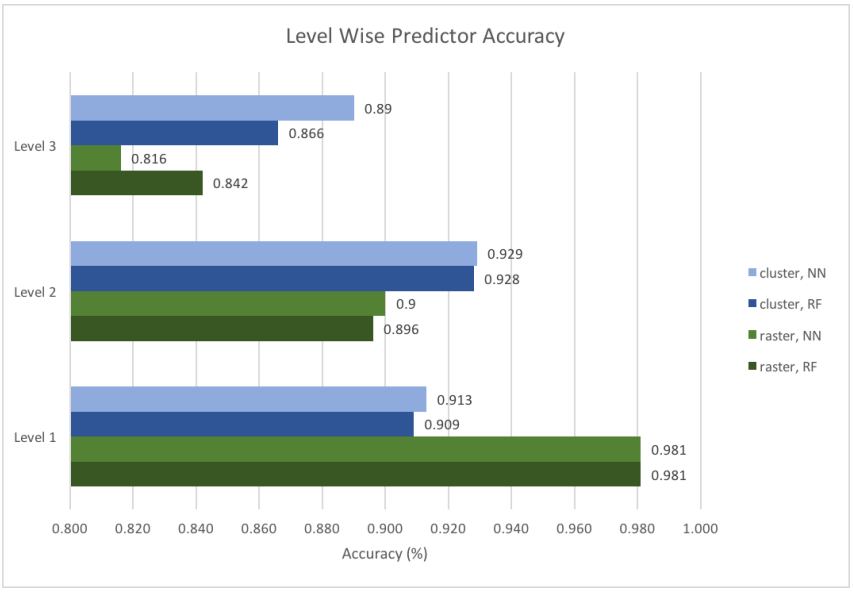


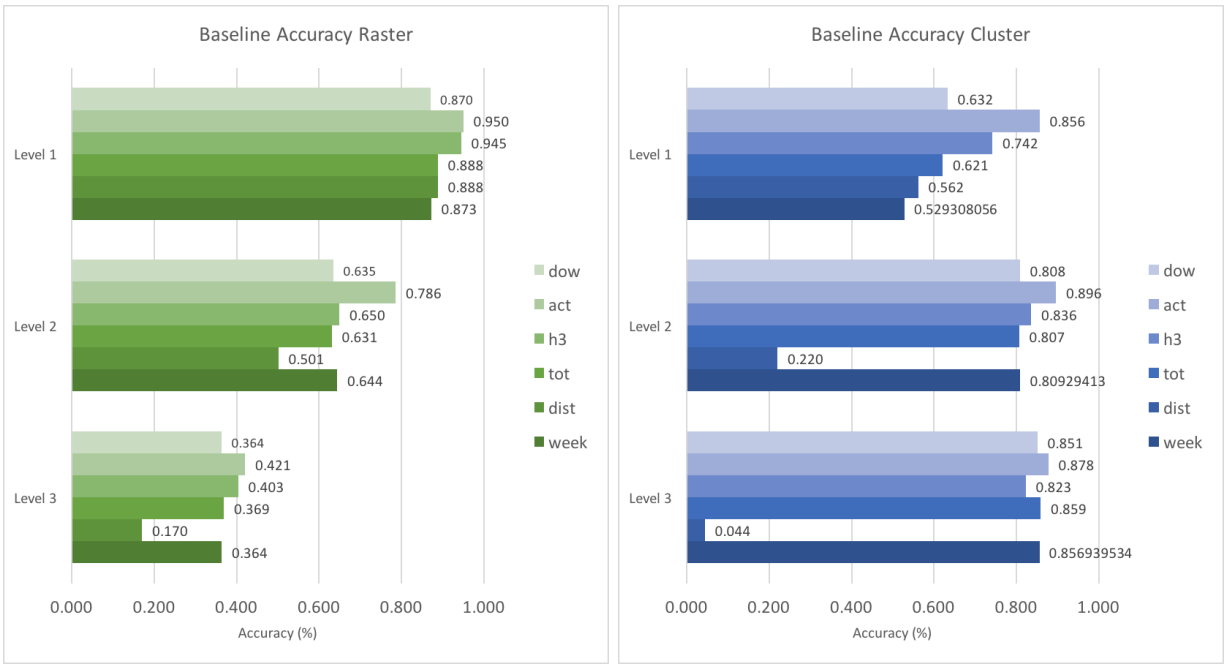

*Figure 5.2: Level-wise comparison of the prediction accuracy for predictors trained with rastered and clustered data*

The prediction accuracy can be broken down to each level of the hierarchy (figure 5.2). Since the prediction system is hierarchical, the next place needs to be predicted correctly at every level to get a correct overall prediction. It can be observed, that the prediction task seems to get trickier for the rastered input, as the accuracy decreases from level one to three. On the other hand, predictions with the clustered input seem to be more difficult in level one than in level two.

Comparing the NN and the RF, it can be observed that they produce similar results when given the same input. In level one and two, the differences lie around one percent. Interestingly, in level three, the NN does better with the clustered input, but RF produces a better result with the rastered input. In general, the NN performs equally or a bit better than the RF on every level with both inputs, except for level three with the rastered input.

Looking at the level-wise baseline accuracy of the rastered input (figure 5.3 a), it can be observed that the baseline accuracy decreases from level one to level three. The pattern looks very similar to the level-wise predictor accuracy, which affirms that the prediction task gets harder with every level. What stands out are the *act* and the *dist* features. The $B_{act}$ performs best on every level. On the other hand, the $B_{dist}$ performs mediocrely on level one, and then it goes down in level two, and in level three the other features perform more than twice as well.
For the clustered input (b), the performance of the baselines is worst in level one, with the exception of $B_{dist}$. $B_{dist}$ again decreases from level to level and therefore only seems to deliver a

good performance on small scales. The other baselines perform similarly in level two and three.




(a) Rastered input

(b) Clustered input

*Figure 5.3: Baseline performance for clustered and rastered input*

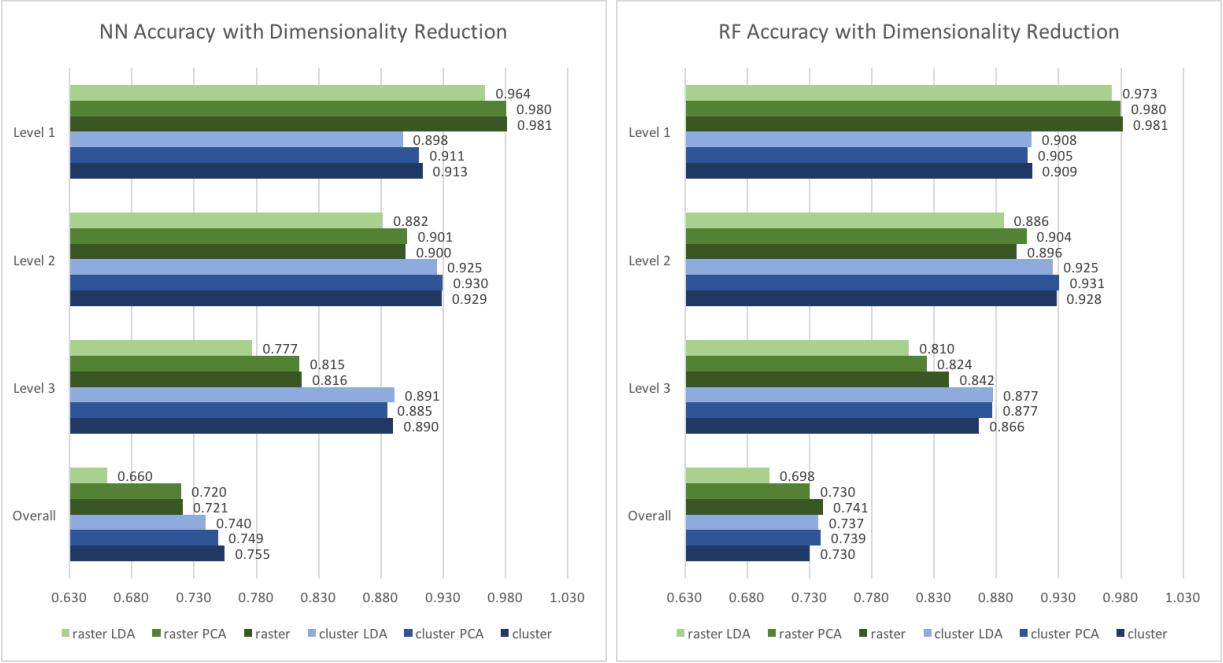
The baselines based on the temporal features all reach a similar accuracy. Thus, it could be argued, that it some of the 700 dimensions in the input vector are redundant.

To reduce the number of dimensions, two different dimensionality reduction algorithms were tested. As indicated in figure 5.4, the employed dimensionality reduction techniques do not improve the prediction in most cases. Overall, the NN performs better with the original input. The results for the RF are a little different. With the clustered input, it performs slightly better if an LDA or PCA is applied. Nevertheless, the differences are marginal (0.7%-0.9%).

Since there are some variations within the each level, the best configuration is chosen for every level (RF vs. NN and LDA vs. PCA vs. original). Table 6 shows the best composition for the clustered and the rastered inputs on every level. Using different inputs and predictors in each level can further emhance the accuracy. For the clustered input the prediction can be improved by 0.3% and for the rastered input by 0.6%, using the best predictor and applying a dimensionality reduction algorithm in cases where it improves the accuracy.

| | Level 1 | Level 2 | Level 3 | Accuracy | Imporvement |
|---|---|---|---|---|---|
| Cluster | NN, original | RF, PCA | NN, LDA | 0.757 | 0.26% |
| Raster | RF, original | RF, PCA | RF, original | 0.747 | 0.64% |

*Table 6: Best prediction accuracy for every level.*



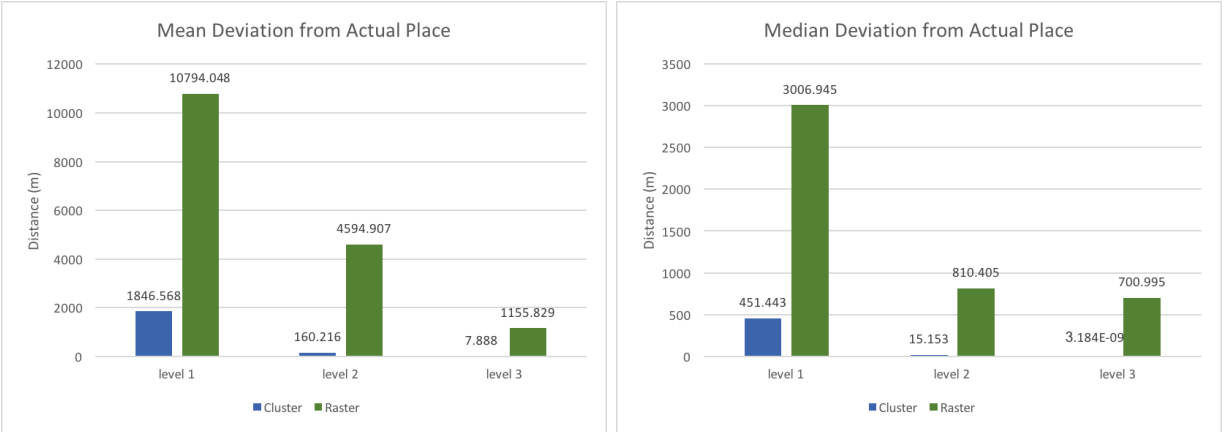

(a) NN dimensionality reduction (b) RF dimensionality reduction

*Figure 5.4: NN and RF performance with dimensionality reduction*

## 5.2 Distance Deviation

Instead of just relying on the predictor accuracy, the distance deviation between the predicted coordinates and the actual place (ground truth) can be used as an additional indicator for rating both inputs.

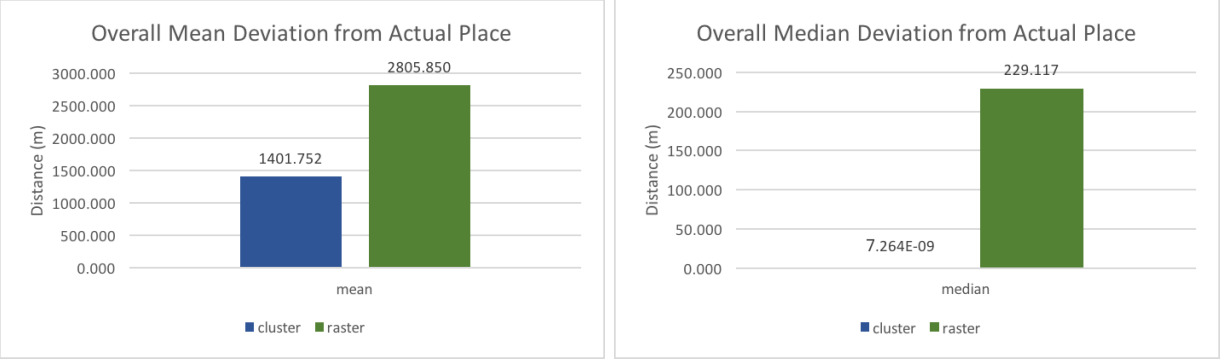
Figure 5.5 shows the mean and the median deviations. As expected the mean and the median decrease from level to level. In general, the mean deviation is much smaller for the clustered input.




(a) Mean deviation (b) Median deviation

*Figure 5.5: Comparison of the distance deviations from the actual place between the clustered and the rastered input. The spatial unit was predicted by the NN.*

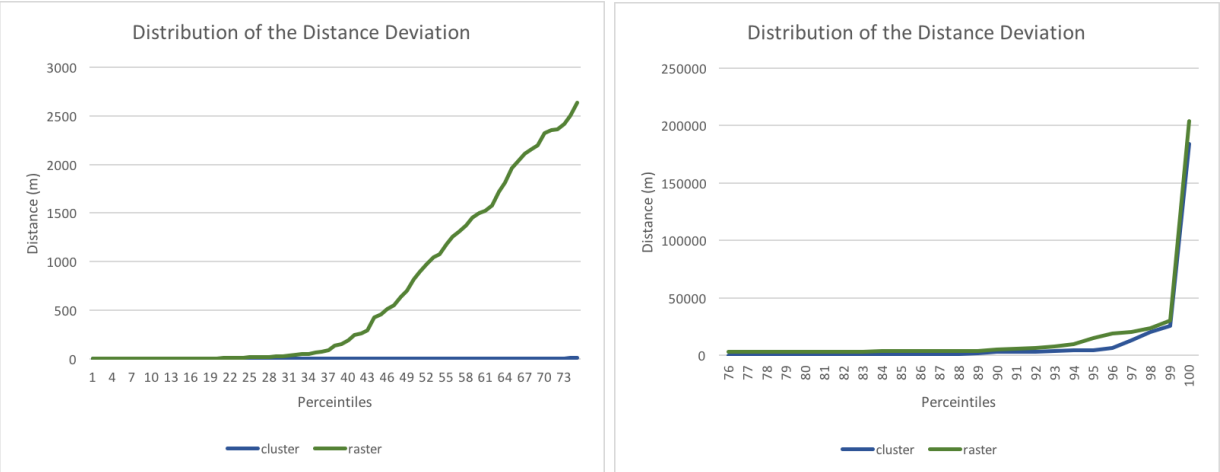
To calculate the overall distance deviation, the distance deviations from wrong predictions in level one, two and three are collected. To this collection, all the distance deviations from correct predictions in the third level are added. Figure 5.6 shows the overall distance deviations. The




(a) Mean deviation (b) Median deviation

*Figure 5.6: Overall distance deviations from the actual place between the clustered and the raster input. The spatial unit was predicted by the NN.*

mean and the median are much smaller for the clustered input, which shows that the model's performance is much better with this kind of input. For 50% of the samples, the distance between the actual place and the prediction is nearly zero meters with the clustered and 820 meters with the rastered input. The small median of the clustered input can be explained with the small area of the clusters in level three. In cases where there are 30 or less points in a cluster in level one or two, these points are treated as clusters in the next level. Thus, if the correct cluster is predicted, the distance deviation is zero. In the second level, 22% of the clusters are represented as a point, and 78% in the third level. This means that the chance is very high that the distance deviation is zero, given that the prediction is correct.




(a) Percentiles 1-75 (b) Percentiles 76-100

*Figure 5.7: Comparison of the percentiles between the raster and cluster input*

As indicated in figure 5.7, the distribution is similar until the first quartile, after that, the distance

deviation of the rastered input increases strongly. The difference between the two inputs is clearly visible in the last quartile as well, which proofs that the clustered input is superior in terms of distance deviation.

## 5.3 User Accuracy

Next to the predictor accuracy and the distance deviation, the user accuracy is also an important measure to evaluate the model performance. Figure 5.8 shows the accuracy of each user. The results were calculated with the NN as predictor and the clustered input. Most of the values are between 60% and 80%, the average is 76.2% and the median 78.3%.

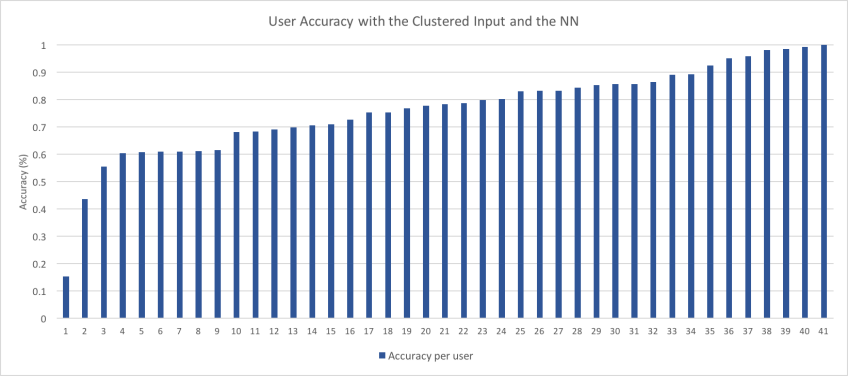



*Figure 5.8: Accuracy per user with the clustered input and the NN*

Particularly striking is user one, with an accuracy of only 15.3%. Looking at the trackpoints revealed multiple possible reasons for the bad performance. First of all, the user visits a lot of different places frequently. He or she commutes to work from Zurich to Bern and visits multiple places on weekends. Most importantly, there is new place that the user started visiting a new place regularly, which is only in the test data, so most probably, the predictor always predicts one of the spatial units with a higher count.

As comparison, user-specific predictors with the same model structure were trained. Figure 5.9 shows the differences between the user-specific and the overall model. Be aware that these accuracies represent the average accuracy of each user and are not weighted according to the number of samples each user has. The best performance is achieved with a user-specific NN trained with the rastered input. The user-specific NN with the clustered input performs second best followed by the user-specific RF trained with the rastered input.
The NN performs better than the RF except for the overall model trained with the rastered input. The NN models single users much better than the RF since for both inputs the accuracy is 4 - 6% higher. In conclusion, it can be said that overall, a user-specific models NN with the

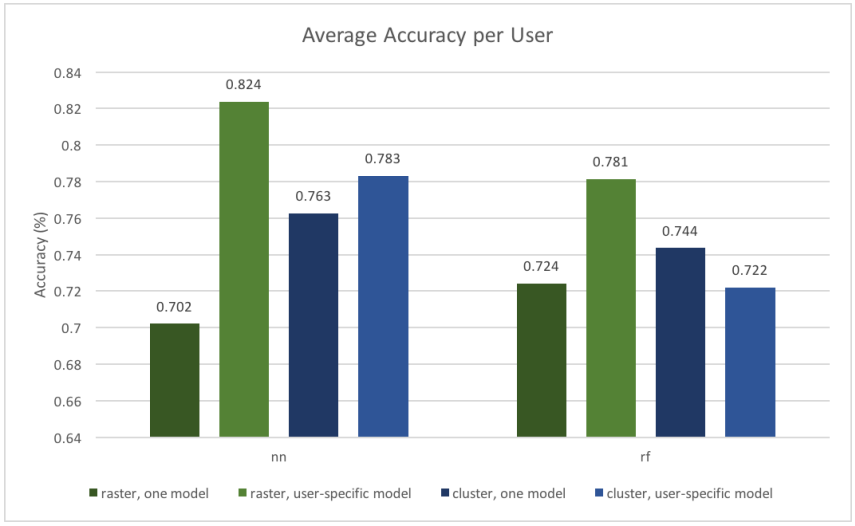
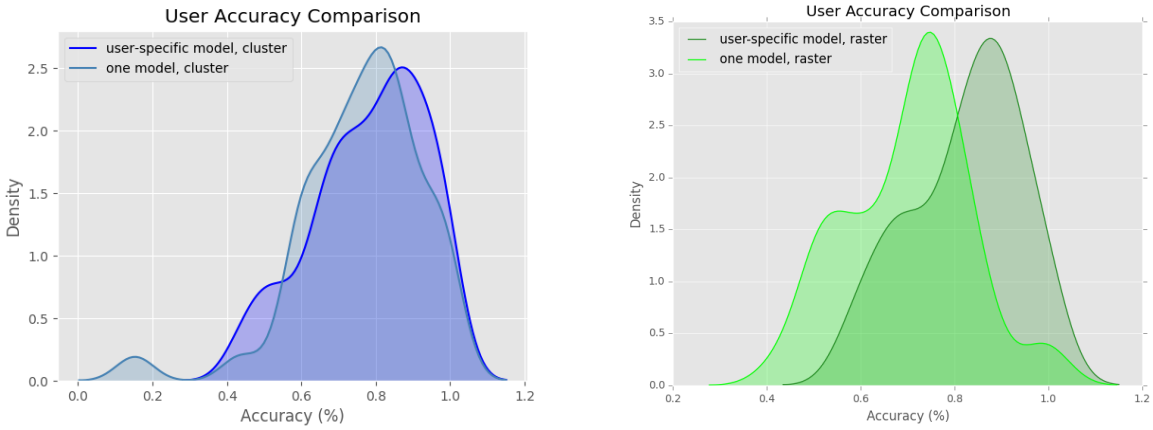



*Figure 5.9: Comparison of the user accuracy for models that are trained with all or only with one user.*

rastered input perform best.




(a) Clustered input

(b) Rastered input

*Figure 5.10: Comparison of the accuracies between the user-specific and the overall model*

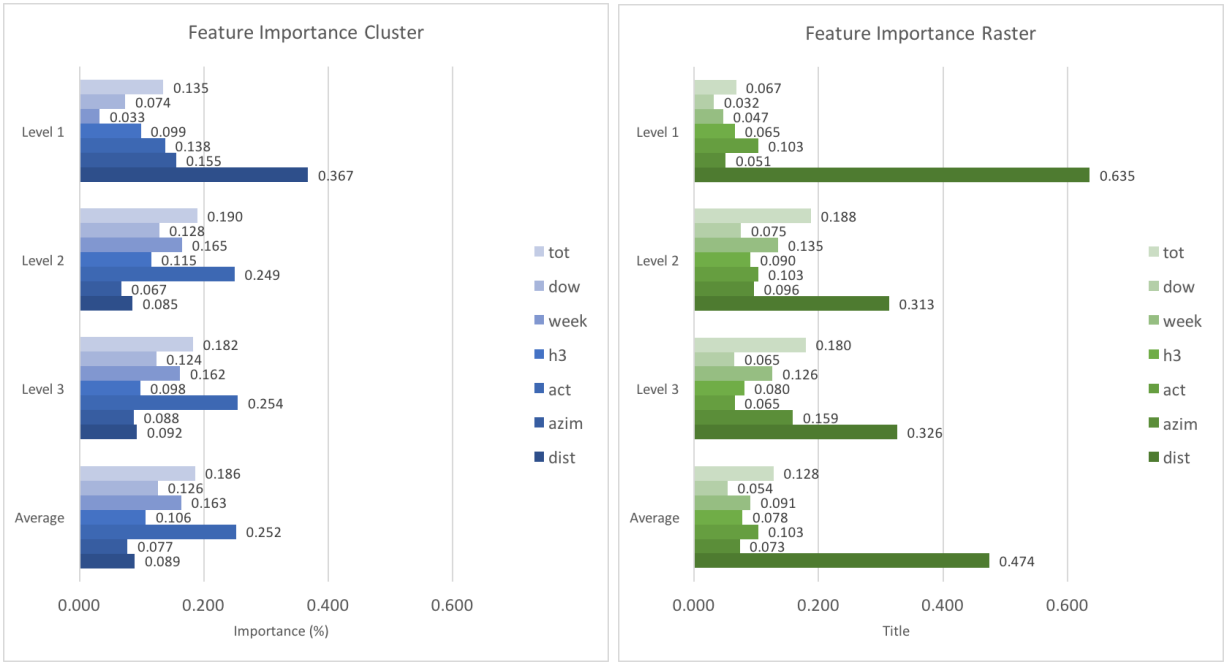
Comparing the accuracies of the user-specific models (figure 5.10) to the accuracies of the overall model, it can be observed that for the clustered input the distributions are relatively similar. The peak of the user-specific model is a bit more to the right. Therefore the probability of having a good accuracy is higher. For the rastered input, the shapes of the distribution look relatively similar, but the user-specific model is shifted to the right.

In all four cases a NN was used. For the clustered input, the overall predictor seems to be able to model all users very good. The user-specific model is only slightly better, which means that the NN is able learn general patterns for all users. With the rastered input it is different. Compared to the user-specific NN, the overall NN seems to have problems to model all users. This shows

that the prediction with the rastered input could be more accurate for the overall NN, but it has problems to find patterns applicable to all users.

## 5.4 Feature Importance

The choice of suitable input features is essential for the prediction. As shown in section 2.5, a random forest can be used to determine the feature importance. Since for every level of the hierarchy a separate random forest is trained, the feature importance can be calculated for every level.




(a) Feature importance of the clustered features

(b) Feature importance of the rastered features

*Figure 5.11: Comparison of the feature importance*

Figure 5.11 shows the feature importance for the clustered and the rastered input. For the rastered input (b) the *dist* feature decreases after level one, but it stays the most important feature in level two and three. The *azim* feature, on the other hand, increases from level one to three. The temporal features are not important in level one, but they increase in the two subsequent levels. Especially the *tot* feature gains much importance. After level one, it is the second most important feature. The *act* and the *h3* feature never reach an importance of more than 11%, which is surprising, because they were the two best overall baseline predictors for the rastered input. It is surprising how important the *dist* feature is. In Level one, it accounts for nearly two thirds, and around one third of the importance in level two and three.

For the clustered input (a), the *dist* feature is only important in level one, where it controls 37% of the decisions. In the next two levels, the *act* feature takes the lead with an importance of around

one fourth. In general, all temporal features gain importance in the two final levels except for the *h3* feature which always stays around 10%. The *azim* feature is the second most important one for the clustered input in level one, but in the second and third level, it gets much less important.

It can be asserted, that the feature importances are much more balanced in the clustered input when compared to the results of the rastered input. The *dist* feature seems to be less significant in the clustered input. Calculating the average over all levels reveals that the *dist* feature is twice as important in the rastered input, where it is the most important feature. For the clustered input, the *act* feature is most important.

Figure 5.12 shows the average importance of the spatial and the temporal features. For the rastered input, the spatial features are on average much more important. For the clustered input, the spatial and temporal features are of equal importance, although it has to be stated that there are four temporal and only three spatial features. Therefore, if the averages would be normalized, the spatial features would be more important.

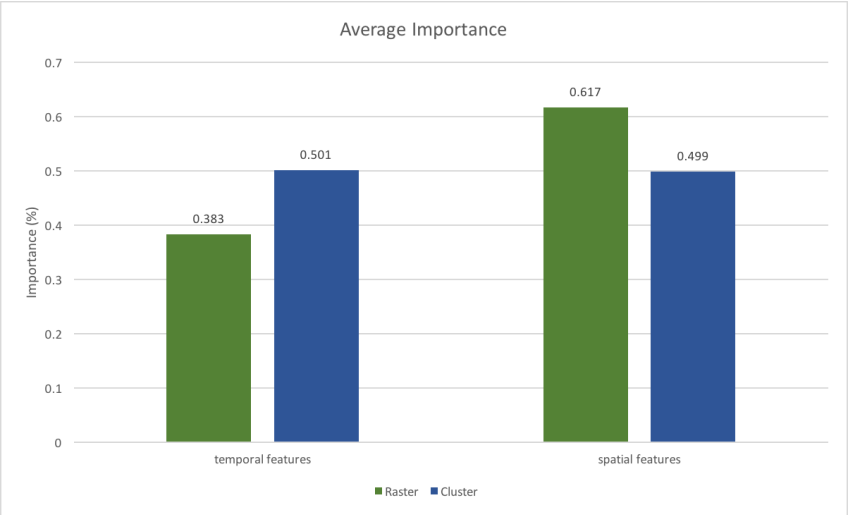
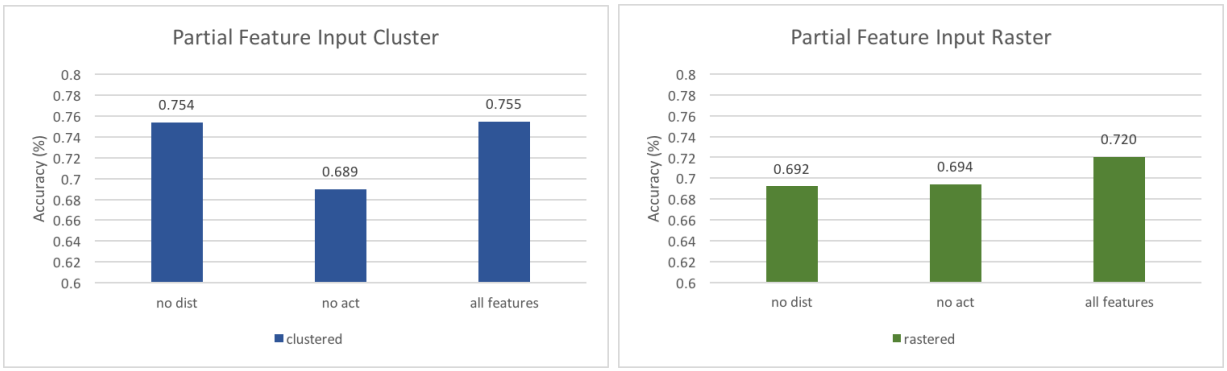



*Figure 5.12: Comparison of the importance of spatial and temporal features. The feature importance of the temporal / spatial features are added up on every level. The average over all levels is used to describe the average importance of the spatial and temporal features.*

While the NN cannot return the feature importance, it can be trained with subsets of features and the results can be compared. To evaluate if the feature importance is also valid for the NN, two models are trained. In the first, the *act* feature was removed from the input, in the second the *dist* feature. Figure 5.13 shows the results from these two models.
Omitting the *dist* feature from the clustered input does not show any effects in the model accuracy. This confirms the findings from the feature importance, since *dist* was assigned an average

importance of only 8.9%. Omitting the *act* feature from the clustered input reduces the model accuracy by 6.6%, which confirms its inportance.




(a) Clustered input (b) Rastered input

*Figure 5.13: Accuracy after omitting the act or dist feature*

Removing the *dist* feature from the rastered input, reduces the accuracy as well, but only by 2.8%. Since its feature importance is at 47.4% the reduction is smaller than anticipated. The accuracy of the NN trained without the *act* feature is reduced by the same value, even though its feature importance is much lower than the one of *dist*.

Given the above mentioned arguments it can be concluded, that the feature importance can be used as indicator, but the absolute values seem to be a bit exaggerated.

# 6 Discussion

## 6.1 RQ 1: Best Model Configuration

To find the best model structure, three main issues must considered. First, the hierarchical structure needs to be discussed and compared to other approaches from previous studies (section 6.1.1). Second, the predictor accuracies need to be evaluated, to find which one should be used (6.1.2). Third, the centroid prediction needs to be evaluated (6.1.3)

### 6.1.1 Hierarchical structure

The goal of this thesis was to predict the next location using only one predictor. The advantages are that there is more training data, and that the already trained model could be used for new users. If a predictor was be trained for every user, a hierarchical structure would not be necessary because the number of possible outputs (places) could be set for every user independently. Other prediction systems typically use only one level (Alvarez-Garcia et al., 2010, Etter et al., 2012), but this is only possible because they only try to predict the next location within a smaller area or because they train a predictor for every user. For instance, in the model of Gambs et al. (2012), a user has up to nine places that can be predicted. They use a user-specific Markov Chain that predicts one of the user's places. De Brébisson et al. (2015) trained one predictor for all users, but the spatial extent of their data was small enough that subdividing it into 1000 clusters was enough. Given the spatial extent of the GoEco! data, many more spatial units are needed to get an accurate decision. Since the data is distributed all over Switzerland, there are numerous entities, and so a direct prediction would be challenging. As there is only one predictor for all users, a common set of possible outputs (spatial entities) had to be defined, and the model had to be able to predict all of these spatial entities. Therefore a hierarchical structure was chosen.

The hierarchical prediction structure ultimately worked well for predicting the next place. As shown in table 7, with the hierarchical approach, a user can have many different places. The proposed model can predict one out of a million cells and one out of 29'270 clusters. A user visited up to 41 clusters and 18 cells in level three. Thus, many more spatial units can be predicted with this approach than in the models proposed by Cho (2016) and Gambs et al. (2012). Therefore, the advantage of the hierarchical structure lies in the fact that users can have varying amounts of places, as opposed to prediction systems where the number of outputs / classes is fixed.
Generally, the hierarchical structure enables the prediction of the next location within a large area and users can have varying amounts of places, which is very important if the same predictor

should be used for all users.

The hierarchical structure chosen in this thesis is not perfect and should be improved further. To be able to compare the results of the clustered and the rastered input, for both inputs three levels were chosen for both inputs. In the first level each model has 100 spatial units, in the second and the third level the clustered has 30 and the rastered input still 100 units. While the prediction with the rastered input on level one seems to be simple (accuracy of 98.1%), the clustered input only reaches an accuracy of 91.3%. Since wrong predictions in the first level usually have a very high distance deviation, the 8.7% wrong predictions in level one have a negative impact on the average distance deviation. Therefore, it should be considered that the prediction task at the first level should be made relatively easy. A simple explanation why the prediction in level one is easier for the rastered input can be seen in table 7. For level one, the average of all visited units per user are counted. For level two and three it can be counted how many spatial units (subunits) the user has visited within the predicted cell. The average of this count shows between how many spatial units the predictor needs to decide, given that it will not choose a spatial unit where the user has never been.
In level one, a user has visited on average 1.7 cells and 3.4 clusters. This means that if by chance, one of the visited cells or clusters would be predicted the accuracy would be higher for the rastered input, because the chance of predicting the correct one is $\frac{1}{1.7}$ instead of $\frac{1}{3.4}$.

| | Level 1 | Level 2 | Level 3 |
|---|---|---|---|
| Cluster | 3.4 | 2.7 | 1.7 |
| Raster | 1.7 | 2.3 | 2.0 |

*Table 7: Average number of places per user and subunit*

Thus, to reduce the difficulty of the prediction in level one, the number of clusters could be reduced, which in turn reduces the average number of visited clusters in level one. To reduce the distance deviation in general, the number of spatial units should be small in the first level and increase afterwards.
Generally speaking, the number of levels and the number of spatial units should be varied to optimize the prediction results.

### 6.1.2 Predictor and Basline Accuracies

The two predictors tested in this thesis performed similarly, with the NN having a slight edge over the RF. Looking at the models trained with all users, the NN reaches the best performance with an accuracy of 75.5%, whereas the best performance of the RF only reaches 74.1% with the rastered input.

With this in mind, it should be stated, that the RF performs better with the rastered input and the NN with the clustered input. That is to say, that depending on the input structure the matching predictor should be chosen. In the proposed model, a RF should be used for the rastered input and a NN for the clustered input.
The baselines perform much better for the clustered input than for the rastered input. This can be explained by the fact that the clusters are self-contained units, which means that the borders of a cluster are defined by areas that nobody has been so far. For the rastered input, the borders of the cells randomly subdivide space. This means that if a user visits the same park multiple times, this park would most probably be contained within one cluster, whereas for the rastered input, it could be randomly split into two cells. Thus, counts that should have been in the same spatial entity because they represent the same place, are subdivided into two cells. This is a serious problem or a baseline that chooses the highest count as the prediction.

Even though the baselines for the rastered input perform much worse, the predictors reach a similar accuracy with the clustered input. Therefore it is highly recommended to use a prediction algorithm for the rastered input, while for the clustered input the best baseline ($B_{act}$) could be used since it is only 8% worse than with the NN. Since the NN outperforms the RF in the user-specific models, it is recommended to use the NN in the future. If a model should be trained with the rastered input of all users, the RF should be used only in level three since it only performs better there (figure 5.2).

Since the length of the input vector is rather high, two dimensionality reduction techniques were tested, namely the LDA and the PCA. In general, they offer little or no improvement on the overall prediction. If dimensionality reduction were used to increase the training and loading speed, PCA would be the technique to go with, since most of the times it gives a better performance than the LDA most of the times. Assuming that computational power is not an issue, on some levels, the PCA or LDA can help to improve the accuracy, but overall, the improvement is so small that it may not be worth the time and effort.

Looking at the literature, there are other predictors that could be used for this kind of problem. Since the input features used in this thesis are similar to the ones used by Etter et al. (2012), a Bayesian network or a gradient boosted decision tree could be tested as well. Then again, Etter et al. (2012) achieved the best results with the NN. Thus the expected results should be approximately the same. The temporal features could be used as temporal transition matrices if the counts would be converted to probabilities. Since Markov chains need states (spatial entities)

and transition matrices, a hidden Markov model could be employed as well. It would be interesting to compare the results, but the findings of Liu et al. (2016) indicate that NN outperform Markov models in the task of next place prediction.According to Liu et al. (2016), Markov chains are constructed based on a strong independence assumption among different factors, that limits their performance.

While in this thesis, a feed-forward NN was used, other research suggests that recurrent neural networks would be a better fit for this task, because they make use of sequential information. In the analysis of De Brébisson et al. (2015), different NN architectures were tested, and while the Kaggle challenge was won with a feed-forward NN, in their custom test, a recurrent NN performed better. With the recurrent NN proposed in Liu et al. (2016), it would be plausible that the accuracy improves. Therefore, a NN with a recurrent structure is another architecture that should be tested in the future.

If multiple predictors were trained, they could be used as an ensemble. Since every predictor may make different errors and one predictor may fail on samples where another excels, different blending strategies should be evaluated to improve the accuracy further. The different predictors could be combined, as shown in Etter et al. (2012) and the ensemble could be used to make the final prediction.

### 6.1.3 Centroid Prediction

After predicting the spatial entity in level three, the user-specific centroid from all places within this unit is calculated and used as the final prediction, as explained in section 4.2.3. The results are accurate because often a user only has one place or multiple places with the same coordinates in one spatial unit. That said, there are other methods that could be used to predict coordinates within a spatial unit. De Brébisson et al. (2015) use the output of the NN to create a weighted average of all cluster centers. The output of the NN is an array of scalar values associated with a spatial unit. Currently, the highest value is chosen to predict the spatial unit and the user's centroid in this unit is the final prediction. In future work, the output values could be used to compute a weighted average for the user based centroids in every spatial unit. This method should help to decrease the mean distance deviation from the actual place, since every time the incorrect spatial unit is predicted, the centroid of correct prediction will be part of the weighted average as well. On the other hand, it could increase the distance deviation for correct predictions, since wrong centroids are part of the average as well.

## 6.2 RQ 2: Data Representation for Input and Output

Before comparing the performance of the predictors with the clustered and the raster input, the differences between the two representations should be highlighted. First of all, the average area of a cell is larger than that of a cluster in every level (figures 4, 3). This causes the total area of the raster to be much bigger than the total area of all clusters. With this in mind, we can look at how many spatial units a user visited on every level.

| | Level 1 | Level 2 | Level 3 |
|---|---|---|---|
| Cluster | 4 | 7 | 12 |
| Raster | 2 | 4 | 8 |

*Table 8: Average number of places per user*

Table 8 shows the average number of spatial units visited by one user. As expected, fewer cells are visited, since the area is bigger and more places fall into one cell. Given that the preditor chooses a visited cell, with the clustered input, one out of 12 clusters (on average) is predicted and with the raster input one out of eight. This increases the the difficulty of the prediction task with the clustered input because there are more clusters to choose from. More importantly, table 7 shows how many units a user visited within the previously predicted cell (one level higher) on average. As described in section 6.1.1, this number influences the difficulty of the prediction task. Thus, comparing how the models perform with the two inputs is biased. However, as the numbers are similar in level two and three, it should be fine to compare these two accuracies between the two inputs and the overall accuracy can be used as an indicator as well.

Since the best accuracy was reached with a NN using the clustered input, it can be argued that it is better suited. The baseline accuracies also indicate that the clustered input is better. With the exception of the $B_{dist}$, all other baselines perform far better in level 2 and 3. The baselines are calculated from the input features, which means that the features in the clustered input are more informative and therefore work better for the task at hand.

The distance deviation shows, that the clustered input is superior because the mean and the median deviation are much smaller (figure 5.6). Since the areas of the cells are bigger, the average distance deviation is also bigger if one of the visited spatial units is predicted randomly. Consequently, if the predictor would work equally well with both input representations, the distances would still be bigger for the rastered input because of its structure. In other words, if a visited cluster or cell is chosen randomly, the prediction will on average be nearer to the actual place for the clustered input. To find the average distance deviation for randomly predicted

spatial units, the average distance between each centroid of the user and the actual place is calculated for every sample (only units that the user has visited are considered as potential predictions). The average of these results (table 9) will be used to normalize the mean distance between the predicted and the actual place.

| | Level 1 | Level 2 | Level 3 |
|---|---|---|---|
| Cluster | 49'347 m | 2443 m | 206 m |
| Raster | 76'125 m | 12052 m | 1402 m |

*Table 9: Average distance between the actual place and the centroids of each user.*

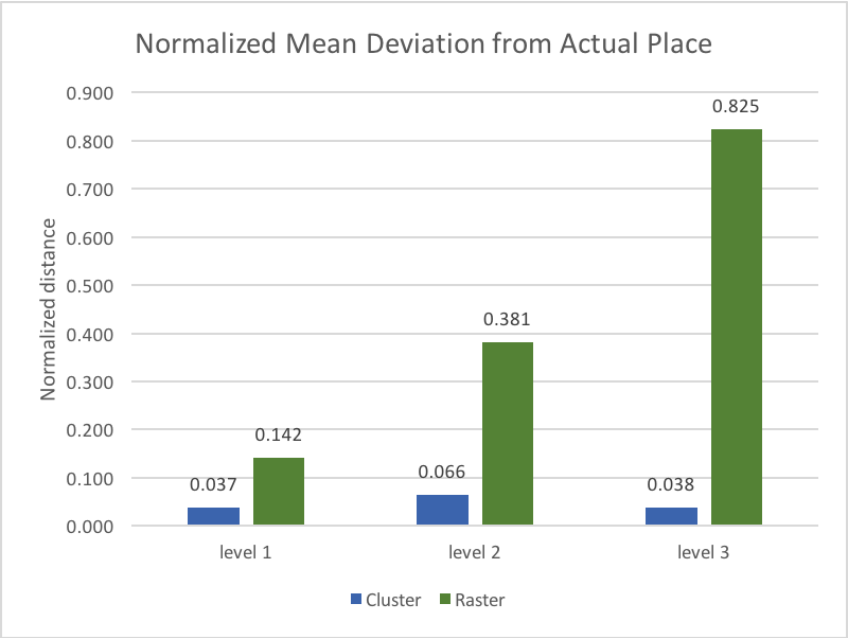



*Figure 6.1: Normalized distance deviation*

By normalizing the mean distance deviation, we can see which model performs better in the prediction task. The normalized distances indicate, that the clustered input performs much better on all levels. Therefore it can be concluded, that the clustered input performs much better according to the distance deviation.

An advantage of the rastered input is that it can predict the next place everywhere in Switzerland. With the clustered input, only places where other users have been can be predicted. For instance, if a user is within a canton where no user has been, with the rastered input the cell where the user is in could be predicted. With the clustered input there would be no cluster, so the prediction would always be far away. Furthermore, if the model should be used in a productive environment where new data is generated, for the clustered input, the clustering and the training of the model needs to be redone. With the rastered input, the structure stays the same, which means that the new data just needs to be preprocessed and then can be used as further training or testing

data.

## 6.3 RQ 3: Input Features

In this thesis, seven different features have been chosen. These features are used in both the clustered and the rastered input. Surprisingly, they perform very differently for the two input types.
Based on the findings in section 5.4, it can be deduced, that with the rastered input mainly the *dist* feature is used for decisions. On the first level, this is not surprising. Since the areas of the cluster and cells are very big, a user mostly stays in the same spatial unit. Therefore the distance feature can be used to predict the nearest spatial unit. Going to the next level, the distances become less important. This is because the area where the prediction can be in decreases, therefore a user does not necessarily go to the nearest spatial unit. While it decreases for the rastered input, it still remains the most important decision factor.

In general, it can be said that the temporal features are less important for the rastered input. While an area which is frequently visited by one user has a good chance of being represented as a cluster, this is not the case for the rastered input . Thus, it is difficult for the predictor to interpret the input. This effect is an example of the modifiable area unit problem (MAUP) described in Unwin (1996) and Wong (2004). In every level of the hierarchy, the points are aggregated into cells of different scales. The rastering produces an arbitrary division of space, which does not consider that very close places could be related, i.e., they belong to the same user. For a user living on a border of a cell, the place visits could be split into both cells. Thus the count in both cells would be half as high than it would be in a cluster. This effect can be seen best when looking at the baseline accuracies as described in section 6.1.2. Even though the same baselines are used, the baselines for the rastered input perform much worse. Except for the $B_{dist}$, the baselines perform nearly double as well for the clustered input, which means, that the values in contained in each feature are more significant for the clustered input.

For the clustered input the *act* feature seems to be the most important one. It is the only feature that contains information from past routes. It reaches the highest baseline accuracy of 67.5% and therefore is nearly as good as the predictors. In the future, this feature could be improved by adding other conditions similar to the ones used for the temporal features (e.g., only use routes recorded on a Sunday).

Ranking all features according to their average feature importance revealed tables 10 and 11.

These tables should be referenced in the future to know which features to use with which input structure and level.

| | level 1 | level 2 | level 3 | average |
|---|---|---|---|---|
| 1. | *dist* | *act* | *act* | *act* |
| 2. | *azim* | *tot* | *tot* | *dist* |
| 3. | *act* | *week* | *week* | *tot* |
| 4. | *tot* | *dow* | *dow* | *week* |
| 5. | *h3* | *h3* | *h3* | *dow* |
| 6. | *dow* | *dist* | *dist* | *h3* |
| 7. | *week* | *azim* | *azim* | *azim* |

*Table 10: Features ranked according to their feature importance from the clustered input*

| | level 1 | level 2 | level 3 | average |
|---|---|---|---|---|
| 1 | *dist* | *dist* | *dist* | *dist* |
| 2 | *act* | *tot* | *tot* | *tot* |
| 3 | *tot* | *week* | *azim* | *week* |
| 4 | *h3* | *act* | *week* | *azim* |
| 5 | *azim* | *azim* | *h3* | *act* |
| 6 | *week* | *h3* | *act* | *h3* |
| 7 | *dow* | *dow* | *dow* | *dow* |

*Table 11: Features ranked according to their feature importance from the rastered input*

Since all temporal features produce similar performances as baselines, it would be possible, that bad performing temporal features would fill in the gap if other temporal features would not be used in the input. The temporal features contain similar data. As the RF chooses the feature that produces the best split, features such as the *tot* could just be slightly better most of the time and therefore features such as the *dow* are seldom included even though they would provide a decent split. As described by Strobl et al. (2007), the feature importance should be used with caution, because the Gini importance, which was used in this case shows a strong bias towards variables with many categories and to continuous variables.
The results given by the NN trained with the partial input indicate that the importance of the *dist* feature is too high. Since the *dist* feature is a continuous variable, it underlines the findings of Strobl et al. (2007).

In future applications, other features should be tested as well. Similar to the transition matrix of a Markov chain, a feature could be generated by looking at the last n places and count which spatial unit was visited how often after these places. Furthermore, for users with few data it might be helpful to extract features based on similar users. For instance, near routes from users that frequent the same spatial units could be used. Since the GoEco! app provides the MOT,

it could be used to extract further features. A very simple one would be to count which spatial units a user typically visits after using the current MOT. Other than that, based on the current position, MOT and movement direction, the destinations of similar routes could be counted.

## 6.4 Limitations

As shown in the previous sections, there are multiple factors that could be improved or need further research. First of all, the hierarchical structure needs further testing. Results from section 6.1.1 indicate that the number of clusters per level should be changed to get a smaller distance deviation. Furthermore, it is difficult to estimate, how the performance would be if the number of levels were changed. With more levels, the prediction in each level is simpler, because there are fewer classes / outputs, but on the other hand, the place needs to be predicted more often. Thus, with every prediction, a part of the samples will be predicted wrongly. In future work, these arguments should be considered to find the optimal balance of the hierarchical structure.

While the overall model performance is decent, experiments showed that user-specific predictors perform better. For a user-specific predictor, the model structure could be simplified by clustering the staypoints of each user and using the clusters as output. With this structure, only one level would be needed, and the performance would be likely to improve even more. Therefore, if accuracy is the most crucial issue, a user-specific predictor should be used. Nevertheless, an overall predictor could be helpful for users with little data, because it learns from other users.
The place definition given in section 4.1.3 reduces the number of staypoints per user. This has a substantial impact on the prediction accuracy. As only staypoints visited more than five times are considered, the prediction is easier since the user is likely to have a regular pattern of going there. Therefore, changing the place definition would have a negative impact on the prediction accuracy. Furthermore, the feature importance could vary with another place definition, because some features might capture the movement patterns better for rarely visited places.

The errors made by the predictors have not been evaluated. Looking at the wrong predictions should help finding situations in which a predictor has difficulties. With this knowledge, new features could be introduced that help solving the problem. Despite having tested seven features, a lot of the contained information is redundant. Therefore other features should be tested.

Last but not least, other predictors should be tested. Hidden Markov chains and recurrent NN are commonly used predictors for this kind of task and could improve the accuracy. After training

multiple predictors, an ensemble of the different predictors could improve the precision even more.

# 7 Conclusion

This work set out to explore how GPS data from mobile phone users can be used to predict the next place. While a lot of the proposed models use user-specific predictors, the challenge in this work was to use one predictor for all users and therefore finding a suitable model structure that allows the model to output the places from all users. To handle a large number of possible outputs, a hierarchical structure was chosen. Places were discretized using two different methods, namely rastering and clustering. The spatial units obtained from these two methods were used to extract features with which a NN and a RF were trained.

The best model performance is achieved by training a NN with the clustered input. With this configuration an accuracy of 75.5% can be reached. Regarding the model configuration, the evidence from this work suggests the following key points:

- Based on the accuracy and the mean / median distance deviation, the clustered input performs better than the rastered input.
- The NN reaches a higher prediction accuracy than the RF.
- The hierarchical structure increases the number of places that can be predicted. Therefore the places can be within a larger area than in comparable literature.
- By calculating the user-specific centroid, the prediction of the NN/RF can be further refined to reduce the distance deviation.
- Using a NN, the best baseline predictor was outperformed by more than 8% suggesting that the predictor learns patterns from the input
- Training user-specific predictors with the same input reaches a higher accuracy than an overall predictor.

In summary, the proposed hierarchical structure with a NN and the clustered input works well in predicting the next place. It can to predict a lot of different places and with great accuracy. If the accuracy is of the essence, the clustered input should be used, even though the raster input performs decently as well. Before deciding which input to use, it should be considered that a model is simpler to maintain with the rastered input because new data can just be preprocessed and fed into the model. For the clustered input, the clustering, the feature extraction and the training need to be done again every time new data is added. While the hierarchical structure proofed to be a success, there is still room for improvement. The number of spatial units and the number of levels should be carefully revised to improve the accuracy further. Although the

user-specific model outperforms the overall model, it is probable that the overall model is helpful for users with little data or new users since it can learn from the other users.

The features used to train the predictors showed different behaviors for the clustered and the rastered input. A model trained with the rastered input bases its decisions mostly on the spatial features while with the clustered input temporal and spatial features are of equal importance. Furthermore, the results showed that the feature importance depends on the level of the hierarchy. On the top level, the geographical distance is the most important decision factor, but after that, the importance of the temporal features increases. For the the baselines, the *act* feature, which looks for past, nearby routes with a similar movement direction, predicts the next location best. For the clustered input it had the highest feature importance while the distance drives most of the decisions in the rastered input. Overall it can be deduced that the feature importance depends on the level but also on the type of input, but since the models perform best with all features, all of them should be used. However, with a different place definition, the feature importance could change.
Training two neural networks with a subset of all features showed that the feature importance should be used with caution. The trends depicted by the feature importance were reflected in the accuracy of the NN, but not as strong as the percentage indicated.

## 7.1 Future Work

Future work should focus on four main categories. First, the other predictors should be tested. Many different predictors and variations of predictors have been proposed, but one that is frequently used is the hidden Markov chain. Thus it would be important to compare it to the performance of the NN. Similarly, a different NN architecture, such as a recurrent NN should be implemented and tested, since they make use of sequential information, which is very important for spatio-temporal data. With multiple predictors, it would be possible to use all of them and create an ensemble predictor. Since some of them may excel where others fail, this could improve the prediction.

Second, the numbers of levels and spatial units per level should be analyzed more thoroughly. More levels lead to less spatial units per level, which simplifies the prediction on each level, but on the other hand, each sample has to be predicted more often, which could lead to a decrease in accuracy. Therefore, the number of levels and the number of units should be varied to find the optimal balance.

Third, other features should be extracted. There are a lot of different features proposed in literature that could be extracted, but there are three main types of features that have not been used in this thesis. Similar to a transition matrix in a Markov chain, the content of features could be based on the last n visited places. The second type would be features based on all or similar users, for example a feature based on the destination of past trajectories from other users that frequent the same spatial units. A third type of feature could be extracted from contextual data. Since the GoEco! app records the mode of transport, a possible feature would be to extract all destinations where the user went in the past while being at a similar position using the same mode of transport. To find which features should be included, the errors made by the predictor should be analyzed. The findings could help to find features that contain the information which the predictor needs to get the correct prediction next time.

And fourth, while the model was trained and tested with only 41 users, in a next step the model should be trained with all 712 users. Having more users will show if the 41 users are representative for the whole sample. Furthermore, the performance of each user should be analyzed, because it could reveal valuable insights that might help to improve the model performance.

# 8 References

Alvarez-Garcia, J. a. et al. (2010). "Trip destination prediction based on past GPS log using a Hidden Markov Model". *Expert Systems with Applications* **37**:12, 8166–8171.

Amit, Yali and Geman, Donald (1997). "Shape Quantization and Recognition with Randomized Trees". *Neural Computation* **9**:7, 1545–1588.

Baumann, Paul, Kleiminger, Wilhelm, and Santini, Silvia (2013). "The influence of temporal and spatial features on the performance of next-place prediction algorithms". *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp '13,* 449.

Bebis, G. and Georgiopoulos, M. (1994). "Feed-forward neural networks: why network size is so important". *IEEE Potentials* **13**:4, 27–31.

Blum, Adam (1992). "Neural networks in C++". *NY: Wiley* **697**:

Breiman, Leo (2001). "Random Forest", 1–33.

Bucher, Dominik, Cellina, Francesca, and Rudel, Roman (2016). "Exploiting Fitness Apps for Sustainable Mobility - Challenges Deploying the GoEco ! App". *4th International Conference on ICT for Sustainability (ICT4S 2016)* September.

Burbey, Ingrid and Martin, Thomas L (2008). "Predicting future locations using prediction-by-partial-match". In: *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments - MELT '08,* 1.

Cho, Sung Bae (2016). "Exploiting machine learning techniques for location recognition and prediction with smartphone logs". *Neurocomputing* **176**: 98–106.

Criminisi, Antonio (2011). "Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning". *Foundations and Trends® in Computer Graphics and Vision* **7**:2-3, 81–227.

De Brébisson, Alexandre et al. (2015). "Artificial Neural Networks Applied to Taxi Destination Prediction".

Do, Trinh Minh Tri and Gatica-Perez, Daniel (2014). "Where and what: Using smartphones to predict next locations and applications in daily life". *Pervasive and Mobile Computing* **12**: 79–91.

Etter, Vincent, Kafsi, Mohamed, and Kazemi, Ehsan (2012). "Been There , Done That: What Your Mobility Traces Reveal about Your Behavior". *Nokia Mobile Data Challenge 2012 Workshop,* 1–6x.

Gambs, Sébastien, Killijian, Marc-Olivier, and Del Prado Cortez, Miguel Núñez (2012). "Next place prediction using mobility Markov chains". *Proceedings of the First Workshop on Measurement Privacy and Mobility MPM 2012,* 1–6.

Gao, Huiji, edu Jiliang Tang, Asu, and Liu, Huan (2015). "Mobile Location Prediction in Spatio - Temporal Context".

Gomes, João Bártolo, Phua, Clifton, and Krishnaswamy, Shonali (2013). "Where will you go? Mobile data mining for next place prediction". *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **8057 LNCS**: 146–158.

Gunduz, Sedef, Yavanoglu, Uraz, and Sagiroglu, Seref (2013). "Predicting next location of twitter users for surveillance". In: *Proceedings - 2013 12th International Conference on Machine Learning and Applications, ICMLA 2013.* Vol. 2. IEEE, 267–273.

Hamerly, Greg and Elkan, Charles (2004). "Learning the k in kmeans". *Advances in neural information processing ...* **17**: 1–8.

Hoang, Minh X., Zheng, Yu, and Singh, Ambuj K. (2016). "FCCF: forecasting citywide crowd flows based on big data". In: *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '16.* New York, New York, USA: ACM Press, 1–10.

Jain, Ashesh et al. (2016). "Structural-RNN: Deep Learning on Spatio-Temporal Graphs". *Cvpr.*

Jing, Yang (2016). "Semantic Place Annotation".

Khoroshevsky, Faina and Lerner, Boaz (2017). "Human Mobility-Pattern Discovery and Next-Place Prediction from GPS Data". In: Springer, Cham, 24–35.

Li, Xutao et al. (2015). "Rank-GeoFM". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '15.* New York, New York, USA: ACM Press, 433–442.

Lin, Miao, Hsu, Wen-Jing, and Lee, Zhuo Qi (2012). "Predictability of individuals' mobility with high-resolution positioning data". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12.* New York, New York, USA: ACM Press, 381.

Liu, Qiang et al. (2016). "Predicting the Next Location : A Recurrent Model with Spatial and Temporal Contexts". *Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016),* 194–200.

Lu, Xin et al. (2013). "Approaching the limit of predictability in human mobility". *Scientific Reports* **3**:

Maind, Ms Sonali B and Wankar, Ms. Priyanka (2014). "Research Paper on Basic of Artificial Neural Network". *International Journal on Recent and Innovation Trends in Computing and Communication* **2**:1, 96–100.

Mandic, D.P. (1995). "Recurrent Neural networks for prediction". *Lancet* **346**:8988, 1501.

Martinez, Aleix M. and Kak, Avinash C. (2001). "PCA versus LDA". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**:2, 228–233.

Noulas, Anastasios et al. (2012). "Mining user mobility features for next place prediction in location-based services". *Proceedings - IEEE International Conference on Data Mining, ICDM,* 1038–1043.

Pankin, M.D. (1987). *Markov chain models: Theoretical background.* Tech. rep.

Siła-Nowicka, Katarzyna et al. (2015). "Analysis of human mobility patterns from GPS trajectories and contextual information". *International Journal of Geographical Information Science* **8816**:May, 1–26.

Sneha, Gandhi and Graham, Benjamin (2014). "Analysing and Predicting Driver's Next Location", 1–11.

Song, C. et al. (2010). "Limits of Predictability in Human Mobility". *Science* **327**:5968, 1018–1021.

Strobl, Carolin et al. (2007). "Bias in random forest variable importance measures: Illustrations, sources and a solution". *BMC Bioinformatics* **8**:8.

Tin Kam Ho (1995). "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition.* Vol. 1. IEEE Comput. Soc. Press, 278–282.

Tran, Le Hung et al. (2012). "Next Place Prediction using Mobile Data". *Proceedings of the Mobile Data Challenge Workshop (MDC 2012).*

Unwin, David J (1996). "GIS , spatial analysis and spatial statistics". **4**: 540–551.

Wang, Jingjing and Prabhala, Bhaskar (2012). "Periodicity Based Next Place Prediction". In: *Procedings of the Mobile Data Challenge by Nokia Workshop in Conjunction with International Conference on Pervasive Computing (Pervasive '12),* 1–5.

Wong, David W S (2004). "The Modifiable Areal Unit Problem (MAUP)". In: *WorldMinds: Geographical Perspectives on 100 Problems.* Dordrecht: Springer Netherlands, 571–575.

Xiong, Liang et al. (2010). "Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization". *Proceedings of the SIAM International Conference on Data Mining,* 211–222.

Xu, Mengwen, Wang, Dong, and Li, Jian (2016). "DESTPRE: A Data-driven Approach to Destination Prediction for Taxi Rides". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing,* 729–739.

Xue, Andy Yuan et al. (2013). "Destination Prediction by Sub-Trajectory Synthesis and Privacy Protection Against Such Prediction".

Yuan, Yihong and Raubal, Martin (2010). "Spatio-temporal knowledge discovery from georeferenced mobile phone data". *Proceeding of the Workshop on Movement Pattern ...* 121–126.

Zheng, Weimin, Huang, Xiaoting, and Li, Yuan (2017). "Understanding the tourist mobility using GPS: Where is the next place?"

Zheng, Yu, Zhang, Lizhu, et al. (2009). "Mining interesting locations and travel sequences from GPS trajectories". *Proceedings of the 18th international conference on World wide web - WWW '09* 49, 791.

Zhou, Changqing, Frankowski, Dan, and Ludford, Pamela (2004). "Discovering personal gazetteers: an interactive clustering approach". *Proceedings of the 12th ...* 266–273.

# Personal declaration

I hereby declare that the submitted thesis is the result of my own, independent work. All external sources are explicitly acknowledged in the thesis.

Place, Date: .....

Signature: .....