Czech Technical University in Prague
Faculty of Electrical Engineering

BACHELOR THESIS

Alexandr Zemek

Prediction of Bicycle Trip Segment Velocities by Machine Learning

Department of Cybernetics

Supervisor of the bachelor thesis:	Ing. Jan Drchal Ph.D.
Study program:	Open Informatics
Branch of study:	Computer and Information Science

Prague, May 2020

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zemek Alexandr**	Personal ID number:	**474714**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Branch of study:	**Computer and Information Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Prediction of Bicycle Trip Segment Velocities by Machine Learning

Bachelor’s thesis title in Czech:

Predikce rychlostí úseků cyklistických tras pomocí strojového učení

Guidelines:

The task is to predict bicycle per-segment velocities based on training dataset composed of GPS track recordings. The prediction model can be, e.g., used in route planning algorithms (which are not part of this work).
1) Familiarize yourself with state-of-the-art approaches of urban bicycle routing and machine learning methods with focus on multiple instance learning paradigm.
2) Develop neural network-based prediction models.
3) Find which features are essential to predict the velocities.
4) Extract additional information from the related map data (Open Street Maps) and use these features to improve the prediction accuracy.
5) Experiment with variable-sized feature vectors employing multiple instance learning methods.

Bibliography / sources:

[1] Using Neural Network Formalism to Solve Multiple-Instance Problems, Tomáš Pevný, Petr Somol, 2016.
[2] Hrnčíř, Jan, et al. "Practical multicriteria urban bicycle routing." IEEE Transactions on Intelligent Transportation Systems 18.3 (2016): 493-504.
[3] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Drchal, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **10.01.2020** Deadline for bachelor thesis submission: ___________

Assignment valid until: **30.09.2021**

Ing. Jan Drchal, Ph.D. Supervisor’s signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt   Student’s signature

Author statement for undergraduate thesis:
I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....
Signature

Dedication

I would like to dedicate my work to my parents who supported me during my entire bachelor's degree study. Also, I would like to thank my supervisor Jan Drchal for his guidance and excellent communication.

Abstrakt

Cílem této práce je vytvořit model strojového učení, který bude schopen predikovat rychlosti segmentů cyklistických tras v Praze a okolí. Model je trénován na datovém souboru složeném z GPS záznamů cyklistických tras a jejich odpovídajících reprezentací v dopravní síti vytvořené z dat Open Street Map. Trasy jsou rozděleny na segmenty, které obsahují různé prvky, jež budou využity pro učení konkrétního modelu. Cyklistické trasy jsou zaznamenány jednotlivými cyklisty a výsledná rychlost vykonaná jednotlivcem je predikována zvlášť pro každý segment dané trasy. Prvky dopravních segmentů jsou extrahovány z dopravní sítě a ty mohou být dále rozšířeny i o další informace z OSM. Po náležité datové analýze se práce věnuje průzkumu různých metod strojového učení, jako jsou například rekurentní neuronové sítě, časově konvoluční sítě nebo regrese náhodných lesů, stejně jako jejich porovnání a vyhodnocení důležitosti dodatečných OSM prvků.

Klíčová slova

predikce strojové učení cyklistická rychlost GPS

Abstract

The goal of this thesis is to build a machine learning model which will be able to predict velocities of bicycle trip segments in the city of Prague and its surroundings. The model is trained on a dataset composed of GPS track recordings and their corresponding representation in a traffic network created from Open Street Map data. The tracks are divided into segments which contain various features which will be used for learning a particular model. The bicycle tracks are recorded by individual cyclists and the generalized velocity performed by an individual is predicted for each segment of the track. The features are extracted from the traffic network and can be extended by even more OSM information. After the proper analysis of the dataset, the work focuses on developing different machine learning methods such as recurrent neural networks, temporal convolutional networks or random forest regression as well as comparing them and evaluating the additional OSM feature importance.

Keywords

prediction machine learning bicycle velocity GPS

Contents

List of Figures **3**

List of Tables **5**

1 Introduction **7**
- 1.1 Motivation . . . 7

2 Problem statement **9**
- 2.1 Related work . . . 10
- 2.2 Problem specification . . . 10
  - 2.2.1 Regression . . . 10
  - 2.2.2 Categorical variables . . . 11
- 2.3 Methods used . . . 11
  - 2.3.1 Neural Networks . . . 11
  - 2.3.2 Recurrent Neural Networks . . . 11
  - 2.3.3 LSTM Networks . . . 11
  - 2.3.4 GRU Networks . . . 12
  - 2.3.5 Temporal Convolutional Networks . . . 13
  - 2.3.6 Random Forest Regression . . . 13

3 Data description **15**
- 3.1 Graph representation of traffic network . . . 15
- 3.2 GPS signal recordings . . . 16
- 3.3 Map-matching . . . 16
- 3.4 OSM data . . . 17

4 Solution approach **19**
- 4.1 Feature extraction . . . 19
  - 4.1.1 OSM feature extraction . . . 20
- 4.2 Data analysis . . . 21
- 4.3 Workflow . . . 23
- 4.4 Project structure . . . 26
  - 4.4.1 Implementation details . . . 27

5 Evaluation **29**
- 5.1 Model comparison with different dataset forms . . . 30
- 5.2 Feature importance . . . 32
- 5.3 Velocity prediction . . . 33

6 Conclusion **35**
- 6.1 Future work . . . 35

References **37**

A Appendix **39**
- A.1 Data analysis details . . . 39
- A.2 Evaluation details . . . 40

List of Figures

2.1 Example of a bicycle track and its segments. Starting point of the track is denoted by the green dot. A black line between two white dots represents a road segment and its origin junction and destination junction. The red line with blue dots is a GPS record of the track. 9
2.2 Repeating LSTM cells and their interacting layers. Source: [1] 12
2.3 Elements of TCN architecture. (a) Dilated convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. (b) Residual block with 1x1 convolution, which is added if the output has different dimension than the input. (c) Connection of residual blocks. Source: [2] 13

3.1 Difference between raw GPS recording (red line) and map-matched track (black line). 17

4.1 Difference between unprocessed GPX (left) and GPX with evenly spaced points added between each two succeeding GPX points (right). 19
4.2 Data analysis of segment numerical features. 21
4.3 Data analysis of segment categorical features. 22
4.4 Data analysis of OSM features around segments. The y axis data are provided in logarithm scale because of prominent zero values. 23
4.5 Measured speeds and times after dataset adjustments. 23
4.6 All tracks in dataset divided into Train set (black), Validation set (blue) and Test set (red). 24
4.7 Example of model learning (left) and testing on a particular track (right). The learning plot shows a progress of training loss and validation loss during the training process. The testing plot shows target times and predicted times for the sequence of segments in the track. The segments are numbered from 0 to the particular number of segments in the track. 26

5.1 Boxplots for each model and each dataset form used in feature importance evaluation. 30
5.2 Boxplots for each model and dataset with and without OSM. 31
5.3 Histogram of absolute prediction errors. 32
5.4 Feature importance evaluated on Random Forest Regression model with 100 weak predictors. The x axis values are provided in logarithm scale. The black line represents standard deviation of measurements. 32
5.5 Random Forest Regression models evaluated on different dataset forms using RMSE of velocity in km/h. 33

A.1 Analysis of OSM features with linearly scaled y axis. 39
A.2 Boxplots for each model and each dataset form used in feature importance evaluation (unscaled error). 40

List of Tables

4.1 Some of the selected models, their parameters, learning hyper parameters and error measured on the first version of dataset. 25

5.1 The neural network models and their parameters with smallest error recorded. 29
5.2 Random forest regression models and their parameters. 29

A.1 Mean and standard deviation values for each model and each OSM feature. 41

1. Introduction

Riding a bicycle is a common means of transport. Unlike cars or public transport, it is a very adaptive way of going from an origin to a destination which is very useful especially in large cities. A person can ride a bicycle not only on a cycleway but also on a road, in a narrow street, through a park, in a forest etc. Most people are also using a bicycle as a workout or a relaxing activity. Another benefit of bicycles is that it is the fastest form of transport without the use of fossil fuels or electricity (not counting the electric bicycles). The bicycle speed is very important for this thesis because we will predict the bicycle velocity values by machine learning algorithms. First of all, we define the problem of predicting numerical values (regression) and propose a machine learning methods used to create the prediction model. This is described in chapter 2 as well as related work research. The data which we are going to work with are described in chapter 3. In chapter 4, we propose a solution to the problem, make a data analysis and describe workflow and implementation. The evaluation of developed models and feature importance is covered by chapter 5. The thesis is concluded by chapter 6.

1.1 Motivation

The learned models and their predictions could be used in a bicycle track planning algorithms. There are a lot of platforms, which can plan the fastest route for cars and other means of transport but they are generally not covering all the possibilities for cyclists. The traffic network for bicycles is more complex than for the other means of transport simply because a cyclist is not as limited as a car driver for example. As was described above, a cyclist has more options where to ride. This thesis might help to cover them and make the decision in choosing the right track easier considering a corresponding velocity.

2. Problem statement

The task of this thesis is to develop a machine learning model which will be able to predict bicycle velocities on a given track ridden along by a cyclist. Each track is recorded by a GPS device and can be visualised on a digital map of a given area. A track is divided into multiple segments where each segment is defined by two junctions and a line between them. The location of starting points, ending points and junctions on the map are given by latitudes and longitudes which fulfill the role of axes in the geographic coordinate system. Each two succeeding segments share one junction which connects them - an ending point of $(i-1)$-th segment is the starting point of i-th segment. This connection creates a sequence of segments which defines a track on a map and gives us the opportunity to extract information about the track from the map data.

The machine learning model should predict a bicycle velocity for each track segment, in other words it should learn a function

$$\hat{y} = f(\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^m$ is a vector of features of a particular road segment and $\hat{y} \in \mathbb{R}$ is the velocity predicted by the machine learning model. If the model uses sequential data to learn, the resulting function is altered to

$$\hat{y}_t = f(\mathbf{x}, \hat{y}_{t-1}),$$

where the predicted value $\hat{y}_{t-1}$ for $(t-1)$-th segment is used as an input for the velocity prediction of t-th segment. The vector $\mathbf{x}$ consists of m components, which describe a particular road segment. These components include numerical data such as length of the segment or maximum allowed speed and categorical data, i.e. road type of surface present at the segment. The model will learn which features are essential for the right

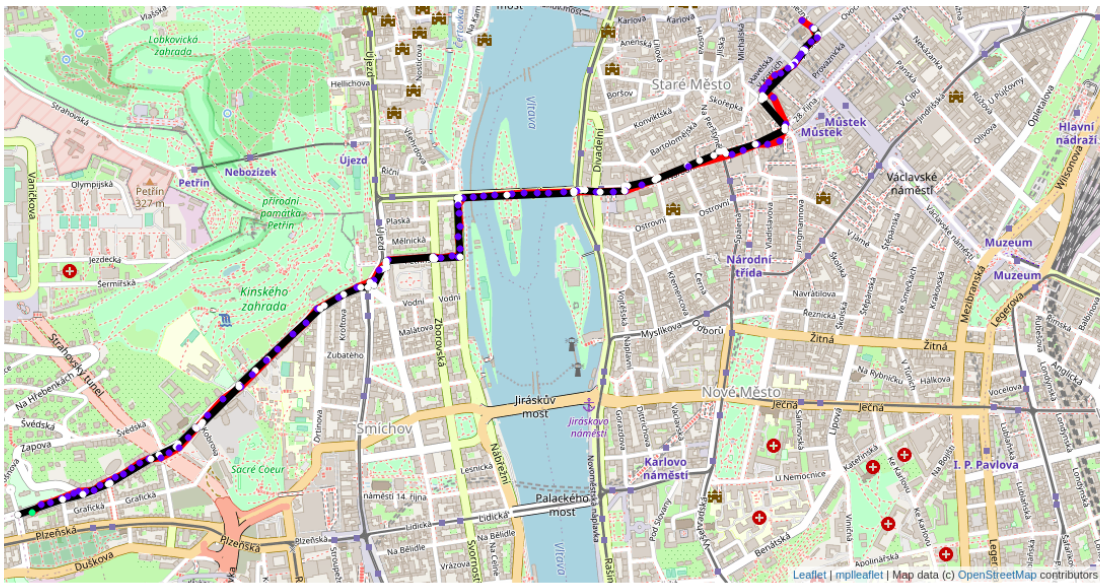

Leaflet | mplleaflet | Map data (c) OpenStreetMap contributors

Figure 2.1: Example of a bicycle track and its segments. Starting point of the track is denoted by the green dot. A black line between two white dots represents a road segment and its origin junction and destination junction. The red line with blue dots is a GPS record of the track.

prediction by assigning bigger weights to the components which are more important. To tell the difference between good and bad prediction, we have a ground truth value $y \in \mathbb{R}$ for each segment. The error is calculated as a difference between predicted values $\hat{y}$ and so called target values y. The smaller the error, the better the prediction.

2.1 Related work

There have been many studies about predicting velocities not just for bicycles but for the traffic flow in general. For example, in [3], authors used a dataset, which consists of measurements performed by over 15000 detectors placed along freeways across California (USA). They then evaluated deep-learning neural network models, which can learn more complex patterns in features contained in the dataset. The use of neural network models for prediction might be helpful, although it is not necessary to implement deep-learning architectures, because we are going to work with much simpler dataset. It will consist of features describing the recorded track instead of features monitoring the situation on road. Furthermore, this work concentrates on velocities performed by an individual rather than the overall traffic flow.

In [4], authors used track data recorded by a GPS device attached to the handlebar of each cyclist. Their tracks are divided into segments and for the prediction, they used the following features: type of facility (segment environment), segment length, number of intersections with traffic signals, average daily traffic, time of day and personal characteristics. Nevertheless, their dataset is quite small (only 8 people were involved in recording their bicycle trips). On the other hand, they were able to use information about a cyclist, which could make the prediction more accurate. The track recordings contained in the dataset which will be used in the implementation were created during an event where many more people participated in. Although the features do not include any data describing cyclist characteristics, the dataset with only track segment features is still large enough to take the neural network approach into consideration. Such methods perform well on a big amount of data.

In [5], authors developed algorithms for bicycle route planning. The bicycle velocity is very significant in this field and the prediction of it might help in calculating their travel time criterion used in their solution. Authors suggested using a traffic web graph created from Open Street Map [6] data. They proposed using segment features such as surface, obstacles like stairs or elevators, places where a cyclist have to dismount the bicycle like sidewalks, cycle infrastructure, road category and crossings like junctions or traffic signals. Our model will use similar representation of traffic network with the use of OSM features to predict bicycle trip velocities. Author of [7] used machine learning algorithms such as Random Forest Regression, Adaboost Regression or Gradient Boosting Regression which were trained and evaluated on similar dataset using a traffic network graph with OSM features.

2.2 Problem specification

2.2.1 Regression

To predict velocity of bicycle track segments, we have to solve a regression task for the purpose of predicting a numerical output. The task of regression, described in [8], is to find a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^m$ for the regression function

$$y = f(x, \boldsymbol{\theta}),$$

where $x \in \mathbb{R}$ is a function input and $y \in \mathbb{R}$ is a function output. In order to find the right parameters, we have to make a set of measurements $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where we have a desired output y_i for each x_i. Then we have to minimize the least squares error, in other words, find a solution to the task

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \sum_{i=1}^{n} (y_i - f(x_i, \boldsymbol{\theta}))^2.$$

2.2.2 Categorical variables

In the dataset, which we are going to work with, there are many features that cannot be represented by a numerical value. We will use categorical variables to represent this information in a numerical form, so that we can use it in regression task. Each categorical variable [9] represents one feature. One-hot encoding is used to convert all possible values into the numerical form.

2.3 Methods used

2.3.1 Neural Networks

This work focuses on machine learning through neural networks. Neural network is a supervised learning system inspired by biological neural circuits which can represent a brain functionality [10]. The system can be viewed as a multiple-layer perceptron as described in [11]. It can learn a function $f : \mathbb{R}^m \to \mathbb{R}^o$ where m is an input dimension o is an output dimension by training itself on a dataset. In our case, the output dimension o is equal to 1 and the dataset which we are going to work with can be represented by a set $T = \{(x_1, \dots, x_m, y)_1, \dots, (x_1, \dots, x_m, y)_n\}$, where $(x_1, \dots, x_m, y)_i$ is a vector of input features $x_1, \dots, x_m$ and target value y for i-th road segment. The network architecture can consist of multiple hidden layers and non-linear activation functions to be able to learn by updating its weights. The predicted features and the target features are compared with a loss function, which calculates the error on predicted data. The goal is to minimise the error by backpropagation and iteration methods such as Stochastic Gradient Descent [12] or Adam algorithm [13]. The predicted function is an approximator for either classification task or regression task. In our case, for predicting bicycle velocity, we need a numerical prediction, so the regression will be used.

2.3.2 Recurrent Neural Networks

Because we are going to work with sequential data, the use of Recurrent Neural Networks might be beneficial. These architectures are using cells stacked together to form a loop [1]. Thanks to the loops, the network is able to remember previous information and use it in the following prediction. However, recurrent networks often suffer from problems such as exploding or vanishing gradients [14]. Therefore, we will use more complex recurrent architectures like LSTM or GRU Networks, which provide a solution to these problems.

2.3.3 LSTM Networks

The Long Short-Term Memory (LSTM) is capable of learning long-term dependencies and thus remembering the previous information for a long time as desrcibed in [1] or [15]. The LSTM cell has two states: a cell state $\mathbf{c}_t$, which accumulates the state information and a hidden state $\mathbf{h}_t$, which is considered as an output of the single cell at a time t. Derived from [16], it consists of four interacting layers where each of them has a matrix of weights $\mathbf{W}$ for input $\mathbf{x}_t$ and hidden state $\mathbf{h}_{t-1}$ and a bias vector $\mathbf{b}$:

- Input gate layer, which takes the current input $\mathbf{x}_t$ and previous output $\mathbf{h}_{t-1}$ and accumulates the cell state:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i).$$

- Forget gate layer, which decides whether to keep or delete the learned information from previous cell state $\mathbf{c}_{t-1}$:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f).$$

- Cell update layer, which transforms the input and previous state to be regarded in in the current state:

$$\mathbf{g}_t = \tanh(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g).$$

- Output gate layer, which scales the output from LSTM cell to be propagated in the hidden state $\mathbf{h}_t$:

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o).$$

Update of the cell state is computed using the gated state and the gated input:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t,$$

where $\circ$ is the Hadamard product. Final output of the LSTM is computed like this:

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t).$$

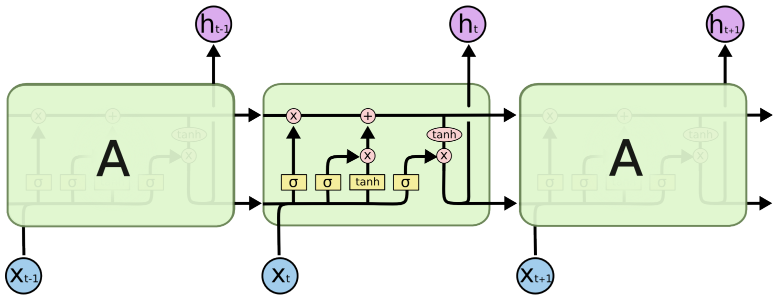


Figure 2.2: Repeating LSTM cells and their interacting layers. Source: [1]

2.3.4 GRU Networks

Gated Recurrent Unit (GRU) Networks are similar to LSTM Networks except for the cell architecture being less intricate. There is no cell state, only hidden state $\mathbf{h}_t$ for each cell is present. Derived from [17], the GRU cell consists of three layers (the bias vector $\mathbf{b}$ was added to keep consistency with LSTM layers described above):

- Update gate layer, which computes how much of the previous information to be kept:

$$\mathbf{z}_t = \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z).$$

- Reset gate layer, which computes how much of the previous information to forget:

$$\mathbf{r}_t = \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r).$$

- Cell update layer, which transforms the input and previous state to be regarded in in the current state:

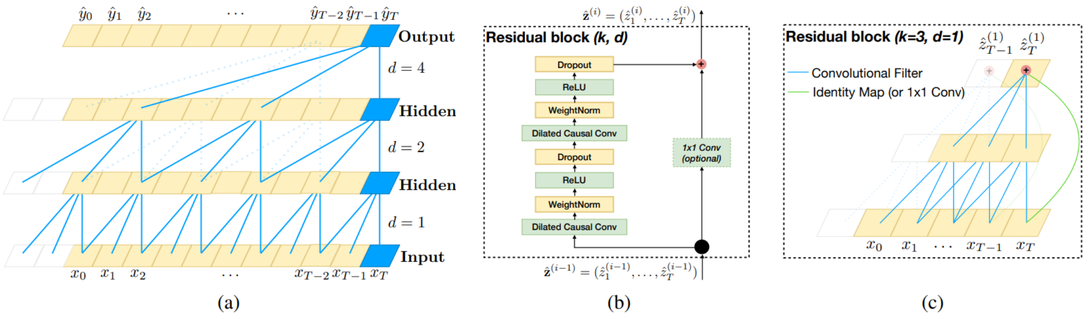
$$\mathbf{h}'_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{r}_t \circ \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h).$$

Final output of the GRU is computed like this:

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \circ \mathbf{h}'_t.$$

2.3.5 Temporal Convolutional Networks

Apart from the architectures mentioned above, we are also going to use and evaluate the Temporal Convolutional Networks (TCN) defined and implemented in [2]. It is a one-dimensional convolutional network, which has shown a great performance in time sequence modelling tasks. Therefore, comparing results with standard neural networks and recurrent neural networks might be helpful in evaluating the learned models.



$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ … $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 … x_{T-2} x_{T-1} x_T
(a)
$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$
(b)
Residual block (k=3, d=1)
$\hat{z}_{T-1}^{(1)}$ $\hat{z}_T^{(1)}$
Convolutional Filter
Identity Map (or 1x1 Conv)
x_0 x_1 … x_{T-1} x_T
(c)

Figure 2.3: Elements of TCN architecture. (a) Dilated convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. (b) Residual block with 1x1 convolution, which is added if the output has different dimension than the input. (c) Connection of residual blocks. Source: [2]

The TCN architecture uses a dilated convolution and a connection of residuals blocks. The dilated convolution operation is defined as

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i},$$

where $\mathbf{x}$ is an input vector, $f : \{0, \ldots, k-1\} \to \mathbb{R}$ is a filter function, k is a filter size and d is the dilation factor. Regular convolution, as used in image classification, is applied when $d = 1$. An illustration of dilated convolution can be seen in Figure 2.3 (a). A residual block contains transformations of the input vector $\mathbf{x}$, which are added to the input of the block itself. These transformations include two layers of dilated convolution, the rectified linear unit (ReLU) activation function, weight normalization and a dropout. An illustration of residual block and its connection is in Figure 2.3 (b), (c).

2.3.6 Random Forest Regression

To tell if the neural network predictions are acceptable, we can make a comparison of the models with other machine learning methods. We chose to evaluate the Random Forest Regressor implemented by scikit-learn library developers [18]. Random Forest Regression as defined in [19], uses combinations of decision trees called forests. The trees are built depending on a random vector $\boldsymbol{\theta}$ and are considered as weak predictors $h(\mathbf{x}, \boldsymbol{\theta})$, which predict numerical output. The final prediction of Random Forest is formed as an average of k weak predictors $h(\mathbf{x}, \boldsymbol{\theta}_k)$.

3. Data description

There are four types of data to work with: A graph representation of traffic network, raw GPS signal recordings of bicycle tracks, recordings of bicycle tracks after map-matching and additional data manually extracted from Open Street Map [6]. The map-matching method is used to map the recorded GPS tracks into the traffic network.

3.1 Graph representation of traffic network

The traffic network was extracted from Open Street Map (OSM) and provided by Ing. Pavol Žilecký. It was extended by information about elevation, which could be essential for speed prediction. The elevation data was taken from Shuttle Radar Topography Mission [20].

Let us define a traffic network in Prague as a directed graph $G = (V, E)$ where V is a set of vertices where each vertex $v \in V$ represents a junction and E is a set of edges where each edge $e \in E$ represents a road segment between two junctions. Each vertex v is a tuple (i, g, g_p, t, t_p, h) which holds information about geographical position of given junction: i is an identifier of the junction, g is a longitude, g_p is a projected longitude, t is a latitude, t_p is a projected latitude and h is en elevation. Projected longitude or latitude means the data is provided in S-JTSK (Ferro) / Krovak[1] reference system. It enables us to compute an euclidean distance between two geographical points by simply calculating the size of difference between two vectors. Each edge e is a tuple $(i_0, i_1, n, m, c, r, s, o)$ which holds information about given segment: i_0 is an identifier of starting junction and i_1 is an identifier of ending junction. Note that the graph G is directed so that for each two edges $e_1 = (a, b, \dots)$ and $e_2 = (b, a, \dots)$ where $a \neq b$ applies that $e_1 \neq e_2$. Element $n \in e$ is a number of car lanes on the road segment, $m \in e$ is a maximum allowed speed in kilometers per hour on the road segment and $c \in e$ is a cycle infrastructure present on the road. It can contain information represented by one of the following values:

- Track - dedicated track for cyclists separated from car traffic
- Zone - pedestrian zone with allowed access by bike
- Lane - mandatory cycle lane
- Sharrow - advisory cycle lane
- None - no cycling infrastructure is present

Element $r \in e$ is a road type which can consist of one of these values:

- Primary - first class roads
- Secondary - second class roads
- Tertiary - third class roads
- Service - generally for access to a building, service station, industrial estate, etc.
- Residential - residential and living streets
- Off-road - roads in countryside, mostly unpaved

[1]http://spatialreference.org/ref/epsg/2065/

- Foot-way
- Crossing
- Steps - stairs
- Cycle-way - dedicated road/street for cyclists
- Unknown

Element $s \in e$ is a road surface which can contain one of the following values:

- Unknown
- Paved Smooth
- Paved Cobblestone
- Unpaved

Element $o \in e$ is a no entry sign, which indicates if it is allowed to travel along this segment. We will use all of the features listed above as inputs for machine learning.

Let us define a cyclist track as a sequence of vertices (junctions) connected by edges (road segments) $(v_0, e_1, v_1, e_2, \ldots, e_{n-1}, v_{n-1}, e_n, v_n)$.

3.2 GPS signal recordings

All of the cyclist tracks are provided by Cycleplanner from a campaign "Do práce na kole"[2] which took place in 2015. There are 1084 different tracks recorded. The recordings are stored in GPX files. It's a special form of XML which consists of tags with timestamps and coordinates (longitudes and latitudes) passed through by a cyclist. These tags create a sequence of cyclist positions in time which can represent a cyclist track. The timestamps will be used later as a target values in machine learning. There might be some errors due to inaccurate signal processing that cause some tracks to be out of a road, which can be viewed in Figure 3.1. Moreover we don't have any information about the track segments from the GPX. That is why map-matching is introduced.

3.3 Map-matching

To map the GPS signal sequence on a road in order to get the right information about the segments on which cyclist rode, we have to use map-matched data. Map-matching is a problem of matching the raw recorded GPS signal into the logical model of the real world [21], i.e. a digital map. There are many forms of the map-matching algorithm [22], which can be categorized into three methods: the incremental (local) method, the global method and the statistical method. Although there are plenty of different implementations, some of them are not fast or precise enough. The map-matching method - Graph Search Based Map Matching - used to create the dataset is introduced and implemented in [23]. It uses a Breadth-first search algorithm with priority queue structure. The process of finding the right nodes to map-match the GPS track is very similar to Dijkstra's shortest path algorithm. The main part of the algorithm can be seen in Algorithm 1.

[2]https://www.dopracenakole.cz

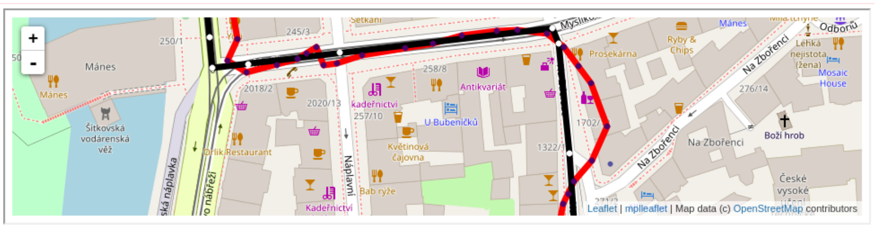


Figure 3.1: Difference between raw GPS recording (red line) and map-matched track (black line).

The data stored in GeoJSON files contains a sequence of junction identifiers and geographical coordinates of a junction. The map-matched tracks are representing the cyclist track in a way that it is possible to extract the information about road segments from the traffic network graph. The difference between track recorded by raw GPS and map-matched track can be seen in Figure 3.1.

Algorithm 1: GSMM algorithm core. Source: [23]

```
Result: Path in the traffic network graph representing the recorded track.
1  init priority queue $Q$;
2  init starting node $n_s$;
3  add $n_s$ to queue $Q$ with priority 0;
4  while $Q$ is not empty do
5      $n_c \leftarrow$ head of $Q$;
6      if $n_c$ not closed then
7          set $n_c$ as closed;
8          if $n_c = c_d$ then
9              backtrack;
10         for $n_n \in$ neighbors of $n_c$ do
11             if $n_n$ not closed then
12                 $c \leftarrow$ computeCosts($n_n$);
13                 add $n_n$ to $Q$ with priority $c$
```

3.4 OSM data

To be able to have more information about surroundings of a track, we can manually extract additional data from OSM. The given dataset is not very large, so augmenting it should improve the learning process, which should lead to a better prediction. Inspired by [5], we have selected the following features along the road segments to extend the basic dataset:

- Number of bus stops
- Number of tram stops
- Number of traffic signals
- Number of trees
- Traffic density (total length of all roads close to the segment)
- Number of sidewalks

4. Solution approach

To solve a regression task, we have to extract information from all available data. The information will be represented by a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{R}^{m}$. Each row of $\mathbf{A}$ represents a particular road segment and each column represents a segment feature. The vector $\mathbf{b}$ is a vector of segment times in seconds. It indicates how long it took for the cyclist to ride through the segment. For simplicity and because we know a distance of each segment, we decided to determine segment velocity by predicting duration per segment in seconds instead of segment speed in m/s or km/h. The velocity in m/s for each segment can be calculated as:

$$v = \frac{segment\ length}{predicted\ segment\ duration}.$$

The matrix and the vector are divided into blocks $\mathbf{A}_i \in \mathbb{R}^{l_i \times n}$ and $\mathbf{b}_i \in \mathbb{R}^{l_i}$, where i stands for track number and l_i is a number of segments in the track i. We use these blocks to distinguish between individual tracks.

4.1 Feature extraction

The features for each track are extracted from traffic network graph, where a sequence of vertices corresponds to a sequence of junction identifiers stored in a particular GeoJSON file representing a map-matched record of a track. For each junction in map-matched sequence, the nearest cyclist position with timestamp extracted from GPX is selected. The target value (time in seconds) for each segment is calculated as a difference between two timestamps.

Nevertheless, the GPS signal is not very reliable and sometimes there are a lot of blind spots. This is probably caused by a temporary signal loss or a malfunction of a record device. This results in inaccurate assignment of GPX timestamps to the map-matched junctions. As we can see in Figure 4.1, there are cases when there are more map-matched junctions in the track than the GPX positions. Specifically in the west area of the track, there are three junctions to which the same timestamp contained in the nearest GPX point will be assigned. We can slightly overcome this problem by thickening the GPX sequence. For each two succeeding GPX points p, q we can generate s evenly spaced points filling up the space over the interval $[p, q]$ considering the cyclist positions as well as the timestamps. Value $s = 5$ is used in the implementation. This heuristics believes that a cyclist passed through a road segment with a constant speed which should not cause problems because the predicted speed will be constant as well.

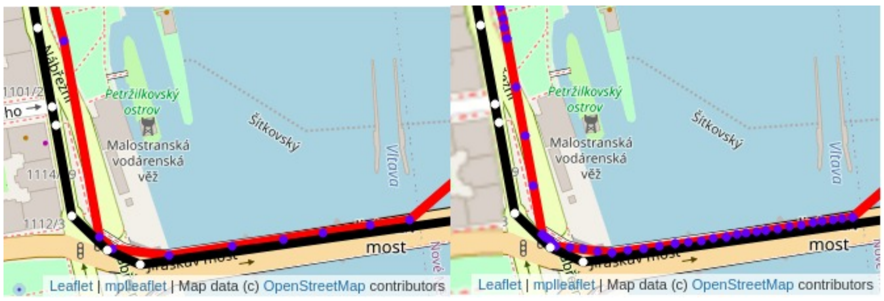

Petržilkovský ostrov
Malostranská vodárenská věž
Šítkovský
Vltava
most
Leaflet | mplleaflet | Map data (c) OpenStreetMap contributors
Leaflet | mplleaflet | Map data (c) OpenStreetMap contributors

Figure 4.1: Difference between unprocessed GPX (left) and GPX with evenly spaced points added between each two succeeding GPX points (right).

Thanks to this we can get more precise target values because each junction will have more options to select the nearest GPX point with the corresponding timestamp.

Unfortunately, there are more errors in the dataset than just a potential GPS signal loss. Some parts of the tracks are map-matched wrongly, which may result in features having i.e. negative time values. There are also cases when a cyclist went through a segment incredibly fast or slow, which might indicate that the cyclist either cheated during the campaign and used other mean of transport or took a break halfway through the segment and forgot to turn off the record device. We have to make a data analysis to point out these problems and suggest what to do with them.

Since all the tracks are located in Prague and its surroundings, there is a possibility that a segment could be a part of multiple tracks. However, each cyclist could ride through the segment with different velocity. These duplicates with different target values might be harmful for the learning process. For that reason and also because we do not have any information about cyclist condition, the target value for each segment is replaced by an average of target values of all segment duplicates.

4.1.1 OSM feature extraction

Now that we have extracted all of the available information, we can still augment the dataset by adding even more features. The OSM features will also be included in the matrix **A**. First of all, we have to download an OSM area of Prague and its surroundings. To save a lot of time and memory, we decided to download and store only the identifiers and coordinates of the OSM nodes, which contain a desired tag. The particular tag is defined as a key-value information, which describes the real meaning of the OSM node. For example, to find a tree in the given area, the tag should contain key-value string '"natural"="tree"'. Again, for memory and time efficiency, each feature (mentioned in 3.4) is stored in a different file in Comma-separated values (CSV) format.

For each road segment s with starting junction a, ending junction b and its latitudes a_{lat}, b_{lat} and longitudes a_{lon}, b_{lon}, the coordinates c_{lat} and c_{lon} of the segment centre c are computed as:

$$(c_{lat}, c_{lon}) = (\frac{a_{lat} + b_{lat}}{2}, \frac{a_{lon} + b_{lon}}{2}).$$

The radius r is computed as a 2-norm of vector $(c_{lat} - a_{lat},\ c_{lon} - a_{lon})$. We say that a geographical point g (respectively OSM node) with latitude g_{lat} and longitude g_{lon} is inside the surrounding of s, if

$$c_{lat} - r \leq g_{lat} \leq c_{lat} + r$$

and

$$c_{lon} - r \leq g_{lon} \leq c_{lon} + r.$$

Because some of the segments are too short and their surroundings are not large enough to include OSM data, we can enlarge the radius r as necessary. In the implementation, the radius r is multiplied by $11 - dist(a, c)$ if $dist(a, c) < 10$, where $dist(a, c)$ is a distance in meters between the starting junction a and the segment centre c. The distance between two geographical points p and q in meters is calculated using the Haversine formula [24]:

$$dist(p, q) = 1000 \cdot 2R \arcsin(\sqrt{h(p, q)}),$$

where

$$h(p, q) = \sin^2(\frac{q_{lat} - p_{lat}}{2}) + \cos(p_{lat})\cos(q_{lat})\sin^2(\frac{q_{lon} - p_{lon}}{2})$$

and R is Earth radius in kilometers. It is multiplied by 1000 to get distance in meters.

We can now count the total number of desired features or calculate the density as a sum of distances between each two OSM nodes within the surrounding of each segment.

4.2 Data analysis

Analysis of numerical features of all road segments in dataset can be seen in Figure 4.2. We can see that there are a lot of nonsensical data. In the first histogram, which shows a cyclist's speed on a segment, there is a noticeable growth at 100 km/h value. We have set this value on segments, where time is equal to zero to avoid division by zero. These values were used in the first version of implementation but they were removed afterwards because it does not make sense to predict values which obviously could not have been achieved by bicycle transport. Average bicycle speed according to [25] is around 19-26 km/h.

In the second histogram in Figure 4.2, we can see that there are not many differences

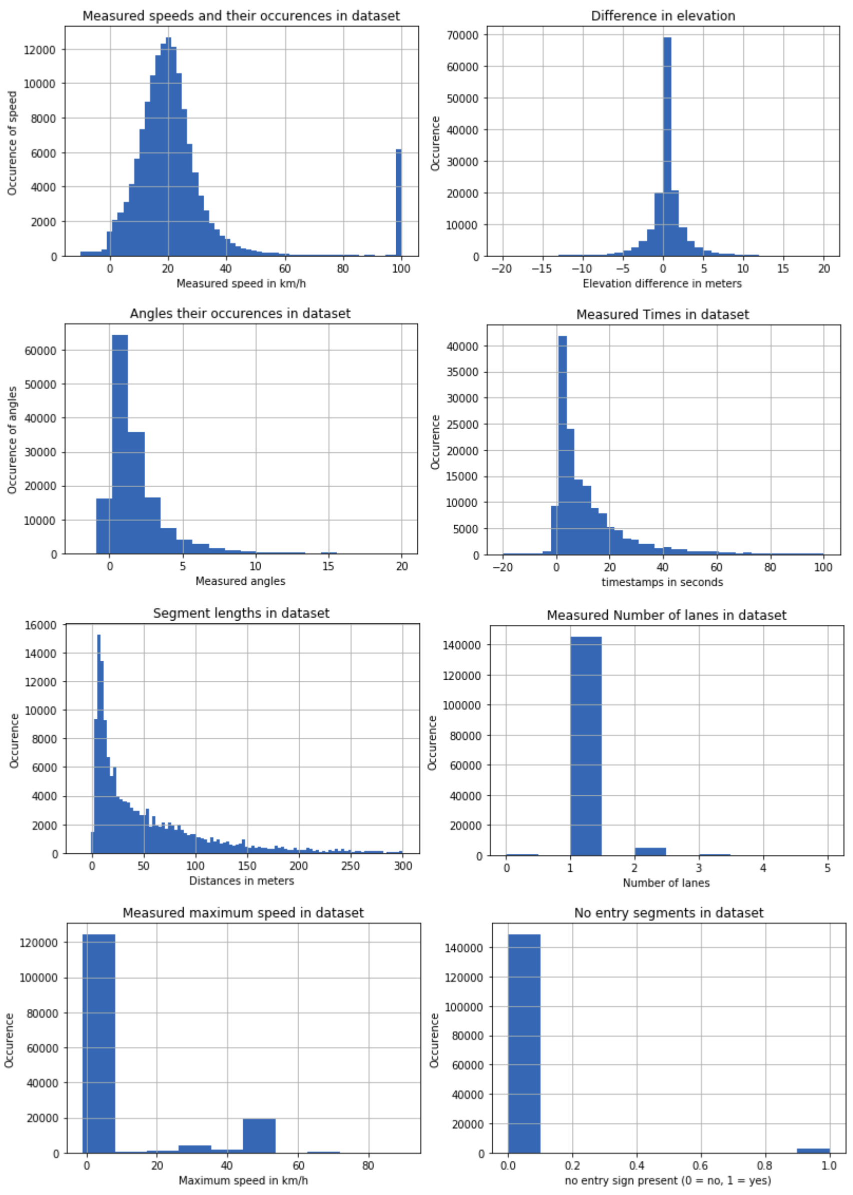


Figure 4.2: Data analysis of segment numerical features.

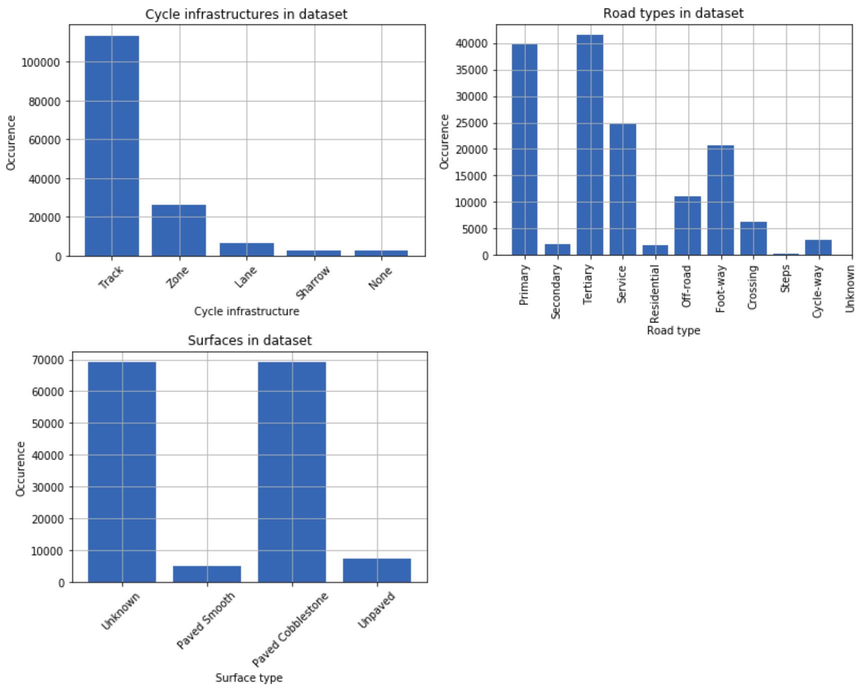


Figure 4.3: Data analysis of segment categorical features.

in elevation between segment's starting end ending point. Most of the values are around zero, which means that cyclists did not ride much uphill nor downhill. The angles of ascent or descent can be seen in the third histogram. In the fourth histogram, we can see that many segments in the dataset are short with regard to recorded times. Unfortunately, there are some negative time values as well. These values should also be removed from dataset. The fact that most of the segments are short is also well-founded in the fifth histogram of Figure 4.2, which shows segment lengths in meters. Majority of segments has only one lane, has no 'no entry' sign and has maximum allowed speed around 50 km/h, although in the seventh histogram of Figure 4.2, there is a growth around 0. This is caused by the fact that there are many missing values which are indicated by value -1.

Analysis of categorical features in the dataset can be seen in Figure 4.3. There are many missing values in surface feature. OSM features are analyzed in Figure 4.4. The majority of values are equal to zero because the dataset consists of many short segments, which have too small surroundings to include any OSM features mentioned above. The histograms y axis data which shows the occurrence of a feature in the dataset are provided in logarithm scale so that we can see the values greater than zero. The histograms with linearly scaled y axis can be seen in Figure A.1 located in appendix.

A machine learning model should not learn features which we consider absurd. The learning process would become complicated and the results would not be as convincing as we desire. To prevent these complications, we can adjust the dataset by removing the road segments of which features are not much trustworthy. That means deleting the road segments with recorded times lesser or equal than 0 and speed above or bellow certain values. Average walking speed of an adult according to [26] is around 3.4-5.1 km/h. Since we are modelling and predicting velocity performed by an adult with a use of bicycle and the dataset contains values around the average walking speed, we decided to get rid of segments with recorded speed lesser than 6 km/h. Maximum allowed speed in Czech cities is 50 km/h and the Figure 4.2 confirms that there are a

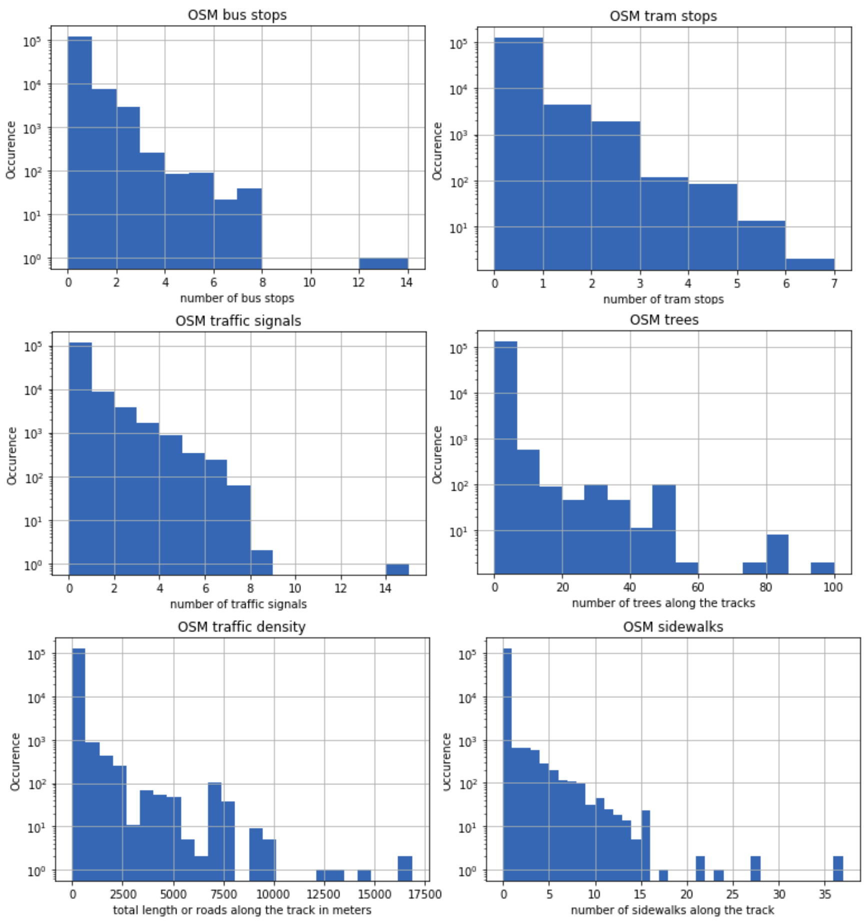


Figure 4.4: Data analysis of OSM features around segments. The y axis data are provided in logarithm scale because of prominent zero values.

lot of segments with this limitation. Although it should not affect an average cyclist much, the dataset contains segments with recorded speed over the limit, which could indicate that a cyclist might have used other mean of transport. For that reason we removed all segments with recorded speed over 45 km/h. The Figure 4.5 shows the speed features and time features after the dataset adjustments.

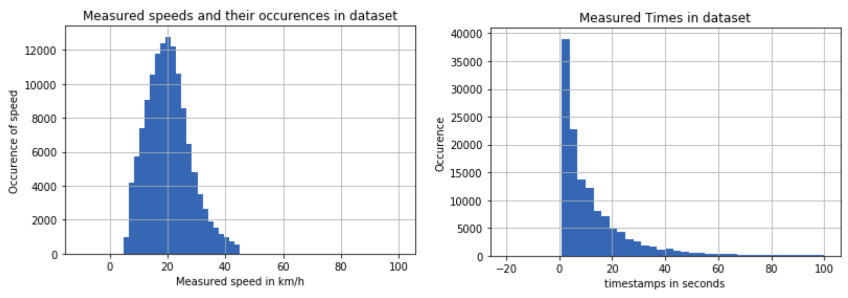


Figure 4.5: Measured speeds and times after dataset adjustments.

4.3 Workflow

After parsing the GPX and GeoJSON data, extracting features from traffic network graph and the data pre-processing, the columns of matrix **A** in the first version

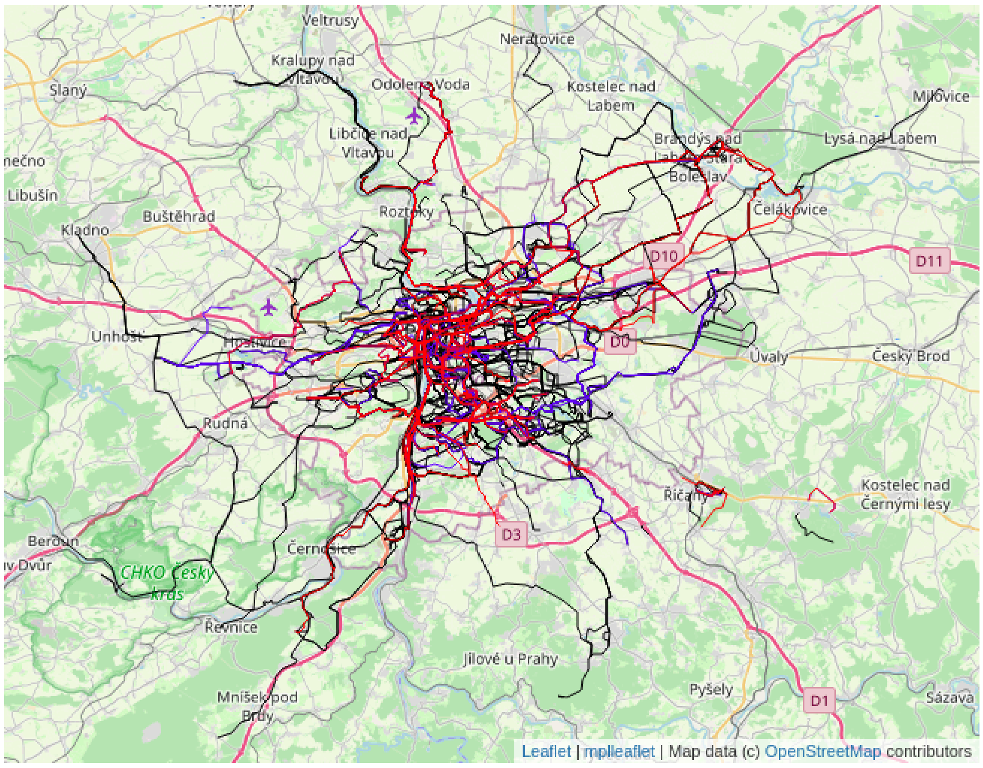


Figure 4.6: All tracks in dataset divided into Train set (black), Validation set (blue) and Test set (red).

contained following features: projected latitude (origin), projected latitude (destination), projected longitude (origin), projected latitude (destination), elevation (origin), elevation (destination), number of lanes, maximum speed allowed, no entry, cycle infrastructure (5 columns), road type (10 columns) and surface(4 columns). The vector **b** contained a target values (segment times in seconds) for each segment. We tried to predict the velocity in km/h and m/s, but the prediction of time in seconds appeared to have better results. The tracks represented by matrix blocks $\mathbf{A}_i$ and target vectors $\mathbf{b}_i$ were randomly divided into three sets: the training set, the validation set and the test set. These sets are disjoint and each of them contains a certain number of tracks. The test set, which is used to evaluate the learned model, contains 100 tracks. The validation set contains 120 tracks and is used in learning process as a test set for each epoch to assess model generalization. The rest of the tracks are contained in the training set. The dataset divided into these three sets can be seen in Figure 4.6.

A good practice for learning a neural network model is to scale the data in order to have an equal unit for each feature. Some of the features are measured in meters, some of the in km/h etc. The data standardization [27] scales the features to have the unit variance and the mean equal to zero. There were some complications with the correct dataset standardization. The correct way is to fit and transform the training set and then transform the validation set and test set with the fitted mean and standard deviation of the training set.

The first developed model did not use sequential data. In other words, the training set and validation set contained shuffled segment features with particular target value. It also did not use the additional OSM data. At first, the model did not learn anything. It was caused by the wrong data standardization and also because we did not make the data analysis and the dataset adjustments mentioned above. After dealing with these problems and experimenting with the model architecture, we came up with a model

Model	Hidden dim.	# r.l.[a]	Dropout	B.size[b]	l.r.[c]	Epochs	RMSE
Basic	[128, 256]	-	[0.2, 0.1]	128	0.001	10000	0.4581
LSTM	128	2	0.5	14	0.0001	10000	0.8173
LSTM	14	2	0.1	14	0.001	10000	0.7793
LSTM	16	2	0.18	14	0.0011	20000	0.5114

[a] number of recurrent layers

[b] batch size

[c] learning rate

Table 4.1: Some of the selected models, their parameters, learning hyper parameters and error measured on the first version of dataset.

with three linear layers: The first layer with an output dimension equal to 128, ReLU as an activation function and a dropout with probability of 0.2, the second layer with the same properties except the output dimension is equal to 256 and dropout to 0.1 and the third output layer with output dimension 1 to have one-dimensional output for each segment. Let us call this model a "Basic model".

To measure how good is the prediction of a model, we use a Root Mean Square Error defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}.$$

The RMSE is calculated using features from the test set. We can use either scaled features or the original features to see the error in desired units. The scaled RMSE of Basic model is 0.4581 and the so called unscaled RMSE is 8.6926.

After some time, we developed a LSTM model, which used sequential data but it was able to learn only with batch size equal to 1. One batch consisted of segment features of one track. However, learning a neural network model with batch size equal to 1 is very ineffective because it has to multiply many small matrices in a row. Learning with bigger batch size and thus multiplying less matrices is faster even though the matrices are larger. Number of segments is different for almost all the tracks and if we want to utilize sequential data with greater batch size, we have to align the $\mathbf{A}_i$ blocks contained in the batch to have the same dimension. The dimension of $\mathbf{A}_i$ is adjusted to $\mathbb{R}^{b \times n}$, where b is a number of segments of the longest track contained in the batch. The rows of matrix blocks of tracks with length shorter than b have to be padded with zero vectors in order to have the same dimension for each block. Therefore, implementing LSTM model with bigger batch size required building a customized dataset, which remembered a number of segments for each track. The tracks contained in one batch had to be sorted by their length and then padded with zero values to fill in the space in the matrix block. The loss function used a mask to filter out the padding values to be able to compute the error only for the desired data. Implementation of this model took too much time because we had a mistake in iteration through the dataset. After fixing this, we came up with a model, which RMSE reached 0.5114 for scaled data and 9.7035 for unscaled data. Some of the models developed and their architectures, parameters and errors can be seen in Table 4.1. These models were learned and tested on Research Center for Informatics cluster[1].

After the LSTM implementation, we focused on augmenting the dataset with OSM data because the error of LSTM model was still bigger than the error of Basic model. We had to adjust the dataset again, because the columns of matrix $\mathbf{A}$ contained projected coordinates but the location of OSM nodes is described by original latitudes and longitudes. The columns of $\mathbf{A}$ in the current version contain following features:

[1]http://rci.cvut.cz/

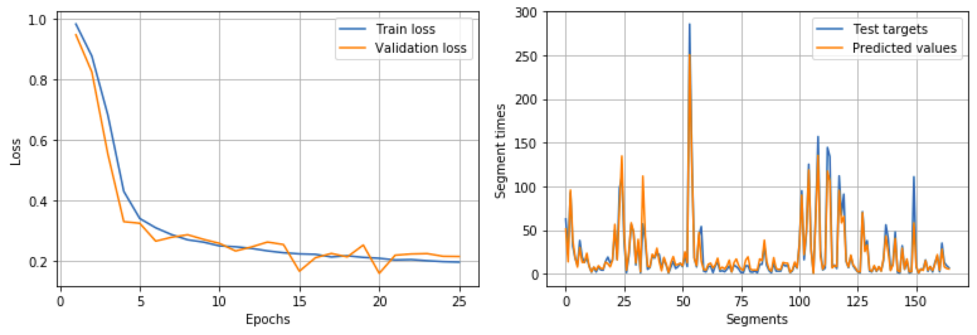


Figure 4.7: Example of model learning (left) and testing on a particular track (right). The learning plot shows a progress of training loss and validation loss during the training process. The testing plot shows target times and predicted times for the sequence of segments in the track. The segments are numbered from 0 to the particular number of segments in the track.

longitude (origin), latitude (origin), longitude (destination), latitude (destination), elevation (origin), elevation(destination), segment length, number of lanes, maximum speed allowed, no entry, cycle infrastructure (5 columns), road type (10 columns) and surface (4 columns). The segment length was added as a separate feature so that the model does not have to calculate the distance by estimating the Haversine formula. The OSM feature extraction was described in subsection 4.1.1. With this new dataset, the error of Basic model was reduced to 0.3512 for scaled features and 6.3677 for unscaled features and the error of LSTM model was 0.3330 for scaled features and 6.0383 for unscaled features. After this, we developed GRU model and a model which use TCN implementation. We took the best architectures of this 4 models and evaluated them in the next chapter.

The next step of this thesis was to experiment with Multiple Instance Learning [28] method, where models use features represented by vectors with variable length divided into bags. The use of variable length features might have been useful in augmenting the dataset with even more OMS features and even further information about OSM nodes themselves. Unfortunately, we did not monitor much of an improvement when OSM data were considered. After the consultation with supervisor, we decided to focus more on comparing the learned models and on evaluation of the OSM feature importance.

4.4 Project structure

All of the scripts and programs were implemented in Python 3.7.3 programming language with the use of Jupyter Notebooks (version 6.0.1). The machine learning models were implemented with PyTorch library. Other libraries and modules used for data pre-processing, visualisation and are listed here:

- *numpy* - Library for matrix and vector operations
- *matplotlib* - Library for visualizing data in plots
- *mplleaflet* - Library used for track visualization [2]
- *gpxpy* - Library for parsing GPX data [3]
- *pandas* - Library for data storage

[2]https://github.com/jwass/mplleaflet
[3]https://pypi.org/project/gpxpy/

- *scikit-learn* - Library for data scaling and Random Forest Regressor implementation
- *OSMPythonTools* - Library for OSM pre-processing [4]
- *osmread* - Library for reading OSM files [5]
- *TCN* - Module with TCN implementation [6]

Implemented scripts and notebooks are listed bellow:

- *bike_speed.ipynb* - Jupyter notebook implementing the workflow
- *data_analysis.py* - Module for data analysis
- *loader.py* - Module for data loading, data parsing, data pre-processing and data visualization
- *network_basic.py* - Module implementing the Basic model
- *network_gru.py* - Module implementing the GRU model
- *network_lstm.py* - Module implementing the LSTM model
- *network_tc.py* - Module implementing the TCN model
- *osm.py* - Module for OSM data pre-processing
- *utils.py* - Module for utility functions

The data, serialized features, learned models and other necessities are separated into the following directories:

- *cluster_logs* - Contains logs of RCI cluster jobs.
- *cluster_models* - Contains *state_dicts* of models learned on RCI cluster.
- *data* - Directory with traffic network graph, track recordings and track map-matching.
- *feat_imp* - Contains results of feature importance evaluation computed on RCI cluster.
- *models* - Directory with *state_dicts* of learned models.
- *npz_files* - Contains serialized features.
- *osm_data* - Directory with downloaded coordinates of OSM nodes.

4.4.1 Implementation details

The main code of implementation is contained in *bike_speed.ipynb* file. It starts with parsing the GPX and GeoJSON files and storing them in *gpx_array* and *geojson_array* variables. The following notebook cells contain track visualisation and data analysis. The dataset is represented by two-dimensional numpy arrays *train_features*, *val_features* and *test_features*. Each of them is accompanied by a list of track lengths (number of segments for each track). The last column of each matrix contains target values. After scaling the matrices and dividing them into inputs and target tensors, we can start a learning process of a model.

The input for Basic model consists of two-dimensional tensor batches of shape *(batch size, dimension)* with randomly shuffled rows of *train_features* and *val_features* matrices. The output consists of vector of given batch size with predicted values for

[4]https://github.com/mocnik-science/osm-python-tools
[5]https://github.com/dezhin/osmread
[6]https://github.com/locuslab/TCN

each segment. The LSTM, GRU and TCN models take a three-dimensional tensor batches of shape *(batch size, length of the longest sequence in batch, dimension)* as an input. The output is a vector of size *batch size · length of the longest sequence in batch.* It consists of vectors of predicted values for each track, which are appended to each other and padded to have equal length.

The models are then tested on *test_features* matrix. The testing is performed on each track individually. The Mean Square Error and Root Mean Square Error both scaled and unscaled are computed for all predicted segment times in each track. The total RSME used in evaluation is calculated on all tracks together. We can then display a plot with target values and predicted values. In Figure 4.7, we can see an example of model training and model testing plots. The following cells in the *bike_speed.ipynb* notebook contain model architecture experiments and feature importance evaluation, which is described in the next chapter.

5. Evaluation

The neural network models which showed the best performance were selected for evaluation. The performance was measured by the Root Mean Square Error during the Workflow process described in 4.3. The architecture parameters such as hidden dimension or number of recurrent layers as well as the learning hyper parameters were adjusted experimentally depending on the final prediction error. The table 5.1 shows their architecture parameters and hyper parameters used for learning. These models

Model	**hidden dim.**	**#r.l.**[a]	**dropout**	**k.s.**[b]	**b. size**	**l.r.**[c]	**epochs**	**RMSE**
Basic	[128, 256]	-	[0.2, 0.1]	-	213	0.001	250	0.3671
LSTM	16	2	0.19	-	14	0.001	250	0.3562
GRU	42	2	0.3	-	19	0.001	250	0.3678
TCN	[11, 11]	-	0.09	4	14	0.001	250	0.3547

[a] number of recurrent layers
[b] kernel size for TCN architecture
[c] learning rare

Table 5.1: The neural network models and their parameters with smallest error recorded.

were also used for evaluating the OSM feature importance to find out how much the velocity prediction improved with the use of OSM data. It is also possible that there would not be any improvement at all or in the worst case the prediction error would increase. Apart from the models mentioned above, we are also going to evaluate two random forest regression models with different parameters described in the Table 5.2. The models were evaluated on the final version of dataset which contains the following features for each road segment:

- Longitude of origin junction
- Latitude of origin junction
- Longitude of destination junction
- Latitude of destination junction
- Elevation of origin junction
- Elevation of destination junction
- Distance (segment length in meters)
- Number of lanes
- Maximum speed allowed (in km/h)
- No entry
- Cycle infrastructure (with 5 categories)
- Road type (with 11 categories)
- Surface (with 4 categories)
- Number of bus stops (OSM)

Model	**number of weak predictors**	**min. number of samples**
RFR 1	100	5
RFR 2	200	2

Table 5.2: Random forest regression models and their parameters.

- Number of tram stops (OSM)
- Number of traffic signals (OSM)
- Number of trees (OSM)
- Traffic density (OSM)
- Number of sidewalks (OSM)

5.1 Model comparison with different dataset forms

The evaluation of machine learning models was executed on Research Center for Informatics cluster. The dataset was adjusted into the following forms: The dataset without OSM data, the dataset with only one OSM feature present and the dataset with all OSM features present. The six models (Basic, LSTM, GRU, TCN, RFR 1 and

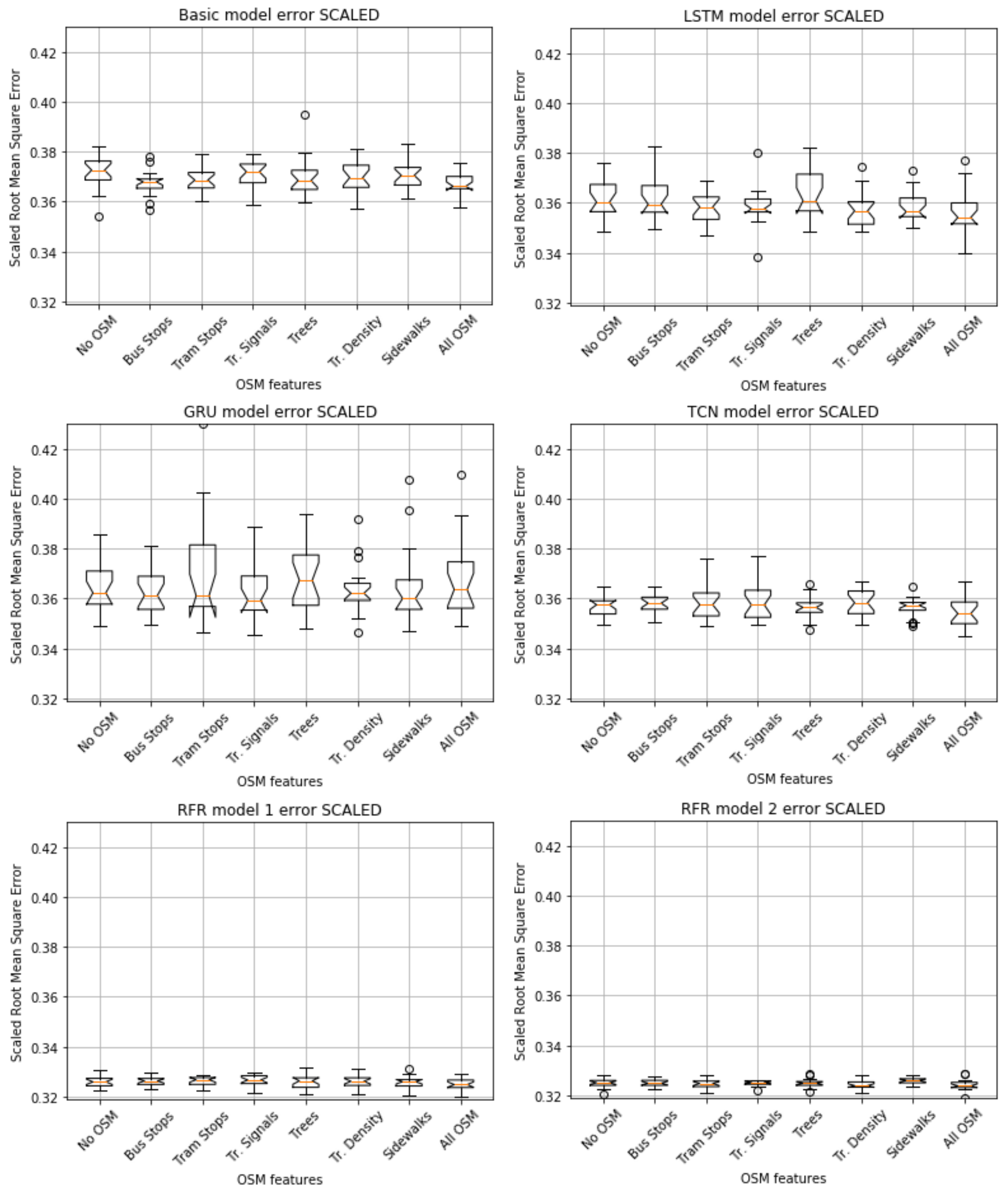


Figure 5.1: Boxplots for each model and each dataset form used in feature importance evaluation.

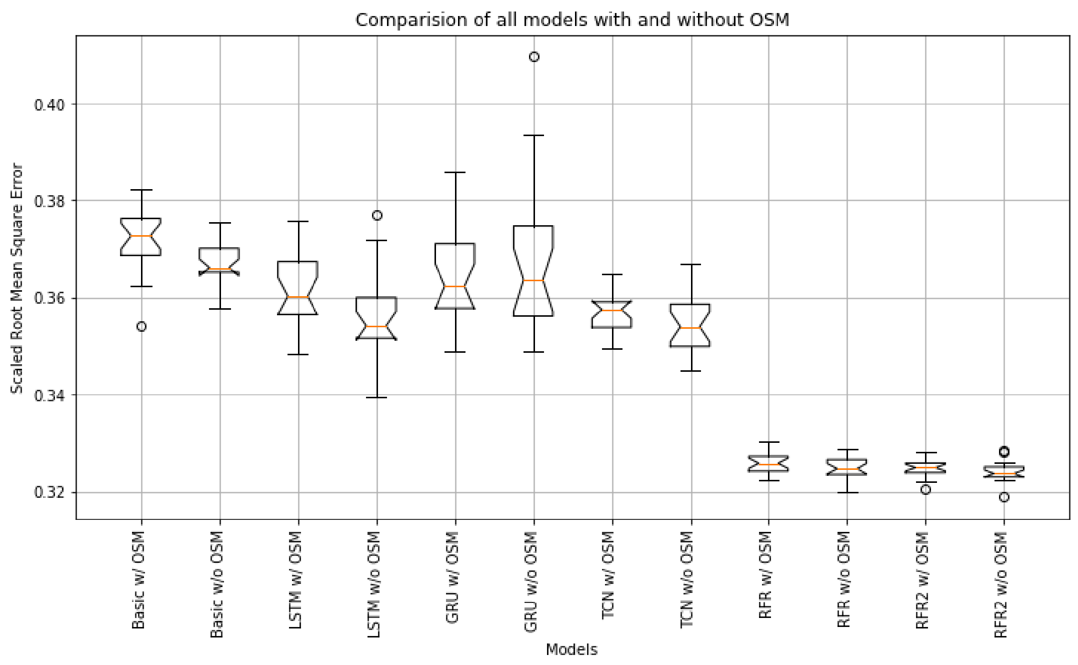


Figure 5.2: Boxplots for each model and dataset with and without OSM.

RFR 2) were trained and tested on twenty measurements for each of the mentioned dataset forms. These measurements were performed with different randomization seeds because the learning algorithm is based on random initialization of the model weights. Because of a big number of measurements, the training algorithm of neural network models was limited to 250 epochs so that the evaluation is finished within 24 hours limit. The evaluation results with scaled errors can be seen in Figure 5.1 and they are represented by boxplots. A boxplot shows statistical information about the measured errors. The orange line represents a median, the horizontal box borders show the first quartile and the third quartile. The places where a box narrows is called notch and it represents a confidence interval. The vertical lines and the dots are representing minimum or maximum values and divergent values. Each model is portrayed in a different plot where each plot shows a boxplot for each dataset form. As we can see, the Random Forest Regression models showed better results than the neural network models. It can be caused by the epoch limitations for network training or the fact that there is still a place for improvement considering the network architecture and its parameters. The LSTM model and the TCN model are better than the Basic model and GRU model. The bigger error of Basic and GRU could be caused by bigger number of neurons which might have caused overfitting. The overfitting means that the model is perfectly trained on training data but fails in generalizing which means that the prediction for previously unseen data is far worse than for the training data. The unscaled error results are represented by boxplots in Figure A.2. The mean and standard deviation of the results can be seen in Table A.1 located in appendix. Although the error on dataset with all OSM data present is lower for almost all models, the difference between the dataset without OSM is not statistically significant. The better view on this can be seen in Figure 5.2. The OSM features slightly improve the prediction but only if all the features are present. The additional features are evidently not as essential as the features extracted from traffic network. The reason why the augmentation of the dataset did not lower the error as much as expected is probably that the tracks are mostly composed of short segments which cannot include the additional extracted data. The fact was already revealed in Figure 4.2 and justified by Figure 4.4 from the

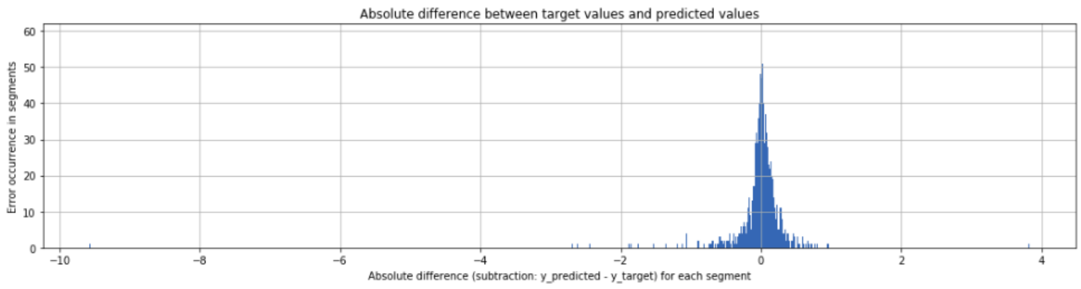


Figure 5.3: Histogram of absolute prediction errors.

data analysis section 4.2, where we can see that most of the values of OSM features are zeros. In Figure 5.3, we can see that the models are predicting values which are often smaller than the target values. The figure shows absolute difference between target values and predicted values for each segment contained in the test set. The absolute error is computed as a subtraction of target value from predicted value. The majority of absolute error is centered around zero which indicates there is no systematic error in predictions. Nevertheless, the lowest RMSE achieved by the Random Forest Regression model is centered around 0.325 which is still quite a large value for prediction error.

5.2 Feature importance

As we can see, the Random Forest Regression models showed better results than the neural network models. We picked the first model with 100 weak predictors and evaluated the importance of all features. The evaluation was executed by a modified

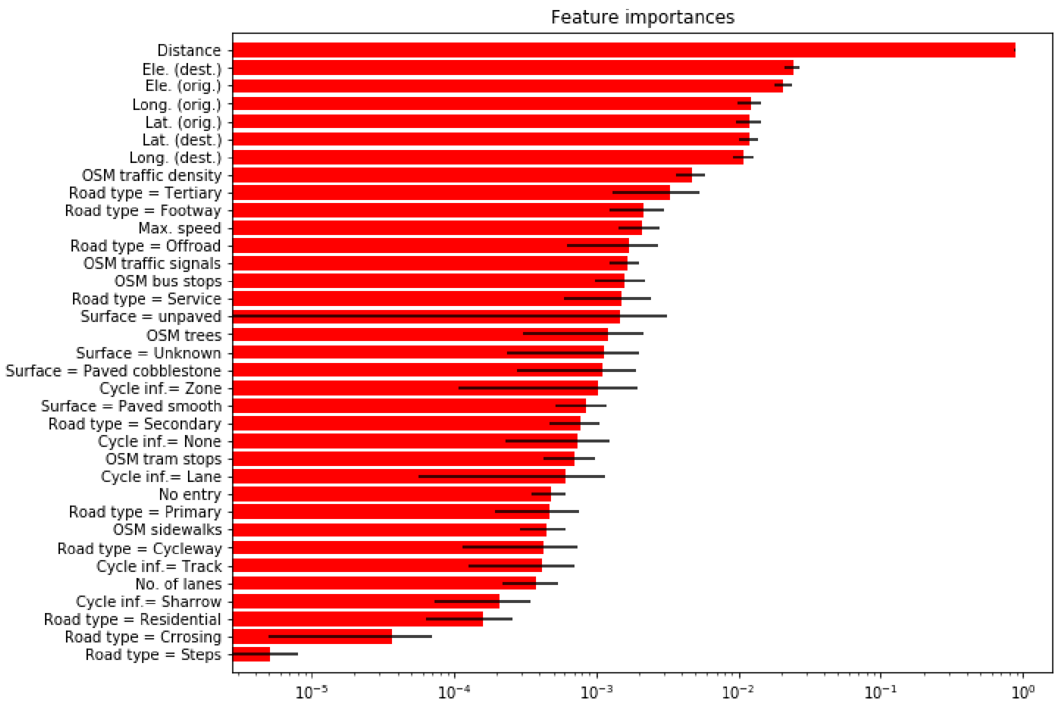


Figure 5.4: Feature importance evaluated on Random Forest Regression model with 100 weak predictors. The x axis values are provided in logarithm scale. The black line represents standard deviation of measurements.

version of the scikit-learn feature evaluation [29], which uses variance as an impurity for each feature to determine how much the feature is important. The script uses the *feature_importances_* attribute of the whole random forest as well as the same attribute of all its trees to calculate the measurement error represented by standard deviation. The results represented with bar plots can be seen in Figure 5.4. Without a doubt, the most important feature for velocity prediction is the distance. Since we are predicting the duration in seconds and not the direct velocity, the distance information is crucial for the prediction. It is obvious that a trip along a longer segment has longer duration than a short segment trip. The coordinates of an origin junction and a destination junction are also essential but not as much as an information about elevation. Even a slight difference between the origin point elevation and the destination point elevation can cause change in the segment duration. If a cyclist rides downhill, the segment trip could be faster. Although it seemed unlikely that the use of OSM features is significant because of very small improvement in prediction, the OSM data (especially the traffic density) appear to have more impact than some of the features extracted from traffic network graph. Is is also possible that the OSM features provide some sort of suitable combination with other features and therefore some of them are more essential than the others. The units representing a feature importance are provided in logarithm scale so that we can see the difference between features which are evidently not as important.

5.3 Velocity prediction

All of the models evaluated are predicting the duration of a segment in seconds. The final velocity prediction is calculated using the segment length in meters and the predicted value. It is multiplied by 3.6 to get the value in km/h which is more common in traffic. The prediction of Random Forest Regression models is the most accurate of all the developed models so we will evaluate the error of the recalculated velocity only for the two models. We can again examine the benefits of additional OSM data by using the different dataset forms. To recalculate the velocity and evaluate the error, we have to use the unscaled data to be able to get the prediction in desired units. The results can be seen in Figure 5.5. The RMSE of velocity is centered between 6.92 and 6.97 km/h. This is rather big error considering the average bicycle speed is around 19-26 km/h which was mentioned in 4.2. There are many reasons why is the overall error so large not just for the Random Forest Regression models but also for the neural network models. The reasons might include problems with map-matching,

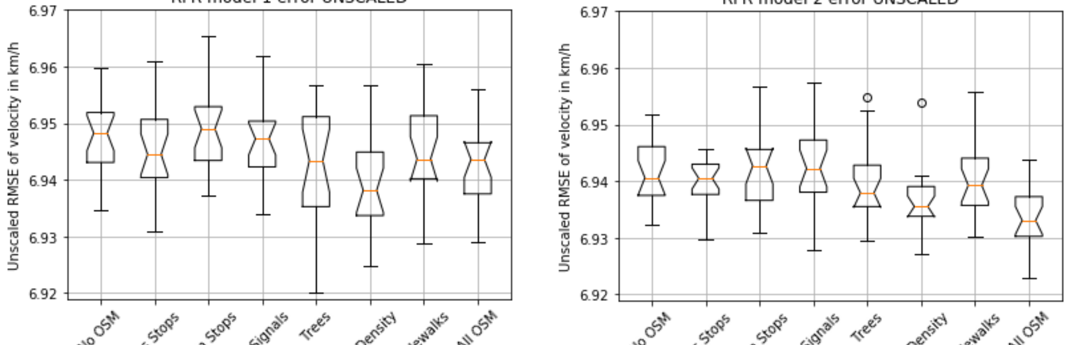


Figure 5.5: Random Forest Regression models evaluated on different dataset forms using RMSE of velocity in km/h.

GPS device malfunction, missing values in the dataset, or even inconveniently chosen network architecture.

6. Conclusion

The thesis assignment was to develop a machine learning model capable of predicting the velocity of an individual cyclist performed on tracks divided into segments. After inspecting the data provided and familiarizing with the state-of-the-art by reviewing the related studies, we implemented the fundamental scripts for extracting the features from the traffic network representation. We then developed the first machine learning model which was trained with the basic features without any adjustments. The prediction was poor because we did not contemplate the possible errors the dataset could contain. After the proper data analysis which revealed the unfavourable information, the bad features were removed from dataset. Thanks to this, we came up with a model which shows a good performance in prediction testing. In order to utilize the sequential data representation of the tracks, we focused on implementing the LSTM networks. We needed to introduce a customized dataset to be able to train a model with larger batches. The training of LSTM network requires a lot of time and a great computing power. For that reason, the LSTM and the other developed models such as GRU or TCN were trained and tested on RCI cluster. The dataset was adjusted for one more time and it was also augmented by additional OSM information to improve the learning process.

We have evaluated different machine learning methods and found out which models were the most suitable for the final prediction. We compared the developed models with other non-neural network based methods. The random forest regression models showed better performance than the neural network models. We discovered that the neural network architectures need more training epochs to learn so that the error would decrease as much as possible. The augmentation of dataset by OSM data was not as essential as expected but some of the extended features appeared to be more important for the prediction than some of the basic features contained in the dataset.

6.1 Future work

The machine learning model could be improved by using even larger dataset with utilization of a more complex traffic network graph. A larger dataset was already provided ten days before the thesis submission. Unfortunately, there were many errors in the track recordings which would require more comprehensive techniques of data pre-processing. With more features, we could consider building a deep-learning neural network architecture which might find more complicated patterns in the dataset. We can also search for a more important OSM features, which might be frequent for most of the road segments. If it is possible to find such features, we should consider using a Multiple Instance Learning method in order to make use of the most information available.

References

[1] C. Olah. Understanding LSTMs. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 19 April 2020.

[2] S. Bai, J. Z. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv e-prints*, page arXiv:1803.01271, March 2018.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[4] A. El-Geneidy, K. J. Krizek, and M. J. Iacono. Predicting bicycle travel speeds along different facilities using GPS data: A proof-of-concept model. *Transportation Research Board*, 2007.

[5] J. Hrnčíř, P. Žilecký, Q. Song, and M. Jakob. Practical multicriteria urban bicycle routing. *IEEE Transactions on intelligent systems*, 18(3):493–504, 2017.

[6] Open street map. https://www.openstreetmap.org/. Accessed: 20 April 2020.

[7] F. Langr. Modelling cyclist behaviour using machine learning, 2016.

[8] T. Werner. Optimalizace. https://cw.fel.cvut.cz/b191/_media/courses/b0b33opt/opt.pdf. Accessed: 24 April 2020.

[9] Simple linear regression - one binary categorical independent variable. https://www.southampton.ac.uk/passs/confidence_in_the_police/multivariate_analysis/linear_regression.page. Accessed: 25 April 2020.

[10] Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network. Accessed: 19 April 2020.

[11] Neural network models (supervised). https://scikit-learn.org/stable/modules/neural_networks_supervised.html. Accessed: 19 April 2020.

[12] Stochastic gradient descent. https://en.wikipedia.org/wiki/Stochastic_gradient_descent. Accessed: 19 May 2020.

[13] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, December 2014.

[14] N. d. Freitas. Machine learning course of department of computer at university of oxford. https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/. Accessed: 19 April 2020.

[15] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv e-prints*, page arXiv:1506.04214, June 2015.

[16] N. d. Freitas. LSTMs for language modelling. https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/practicals/practical6.pdf. Accessed: 19 April 2020.

[17] S. Kostadinov. Understanding GRU networks. `https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be`. Accessed: 19 April 2020.

[18] Random forest regressor. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`. Accessed: 6 May 2020.

[19] L. Breiman. Random forests. In *Machine Learning*, volume 45, pages 5–32. Kluwer Academic Publishers, 2001.

[20] Shuttle radar topography mission. `https://www2.jpl.nasa.gov/srtm/`. Accessed: 1 May 2020.

[21] Map matching. `https://en.wikipedia.org/wiki/Map_matching`. Accessed: 20 April 2020.

[22] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, page 352–361, New York, NY, USA, 2009. Association for Computing Machinery.

[23] D. Fiedler, M. Čáp, J. Nykl, P. Žilecký, and M. Schaefer. Map Matching Algorithm for Large-scale Datasets. *arXiv e-prints*, page arXiv:1910.05312, September 2019.

[24] Haversine formula. `https://en.wikipedia.org/wiki/Haversine_formula`. Accessed: 1 May 2020.

[25] H. Reynolds. How to cycle faster and increase your average speed. `https://www.cyclingweekly.com/fitness/training/13-ways-increase-average-cycling-speed-144937`. Accessed: 2 May 2020.

[26] E. Cronkleton. What is the average walking speed of an adult? `https://www.healthline.com/health/exercise-fitness/average-walking-speed`. Accessed: 2 May 2020.

[27] scikit-learn Standard scaler. `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`. Accessed: 4 May 2020.

[28] T. Pevný and P. Somol. Using Neural Network Formalism to Solve Multiple-Instance Problems. *arXiv e-prints*, page arXiv:1609.07257, September 2016.

[29] Feature importances with forests of trees. `https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html`. Accessed: 19 May 2020.

A. Appendix

A.1 Data analysis details

Figure A.1 shows analysis of OSM feature in linear scale. We can see that the majority of feature values is equal to zero which is caused by the fact that there are many short segments in dataset, which cannot include any other features within its surroundings.

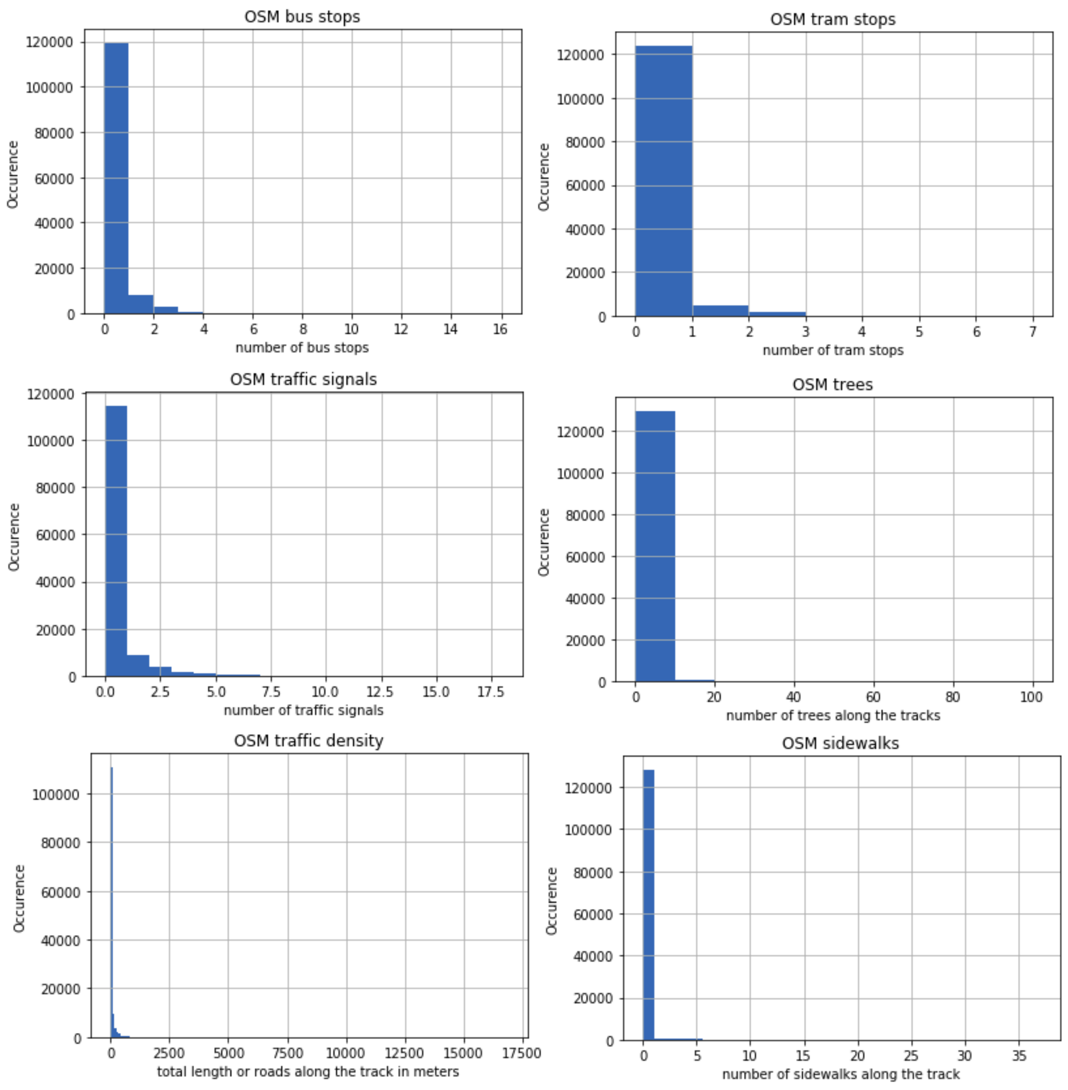


Figure A.1: Analysis of OSM features with linearly scaled y axis.

A.2 Evaluation details

Figure A.2 shows evaluation of model comparison with different dataset forms in unscaled units to be able to have an idea about the error in seconds. The Table A.1 shows mean values and standard deviation values for the evaluation measurements.

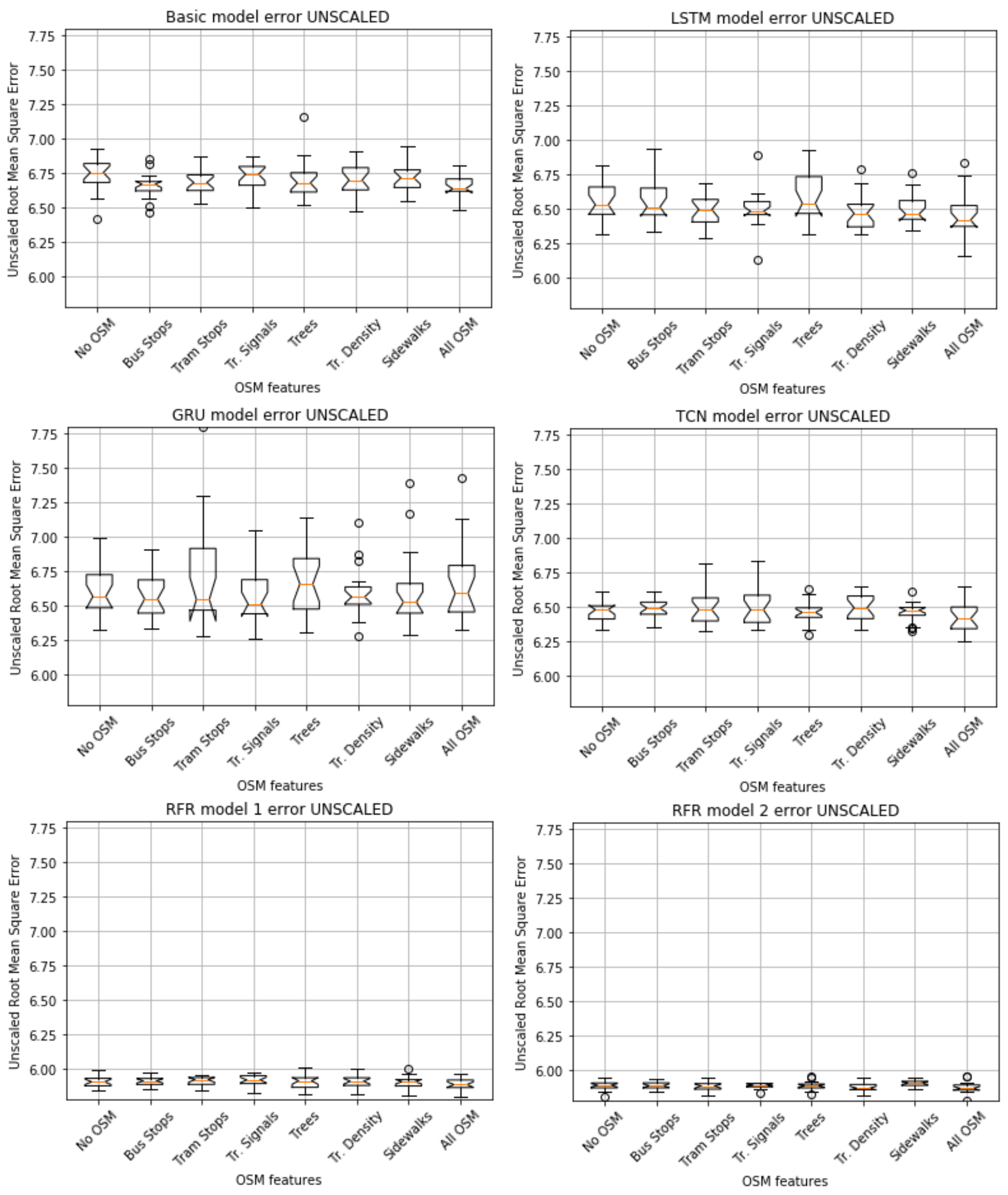


Figure A.2: Boxplots for each model and each dataset form used in feature importance evaluation (unscaled error).

Model	**OSM data**	**Scaled error**		**Unscaled error**	
		Mean	Std	Mean	Std
Basic	None	0.3719	0.0067	6.7431	0.1209
	Bus stops	0.3675	0.0048	6.6623	0.0866
	Tram stops	0.3689	0.0049	6.6885	0.0882
	Traffic signals	0.3712	0.0053	6.7293	0.0969
	Trees	0.3699	0.0077	6.7067	0.1404
	Traffic density	0.3699	0.0059	6.7062	0.1071
	Sidewalks	0.3704	0.0060	6.7151	0.1089
	All	0.3671	0.0044	6.6554	0.0798
LSTM	None	0.3615	0.0081	6.5547	0.1471
	Bus stops	0.3616	0.0085	6.5556	0.1538
	Tram stops	0.3582	0.0059	6.4949	0.1077
	Traffic signals	0.3587	0.0074	6.5038	0.1350
	Trees	0.3631	0.0093	6.5841	0.1682
	Traffic density	0.3577	0.0068	6.4854	0.1231
	Sidewalks	0.3588	0.0062	6.5046	0.1121
	All	0.3562	0.0093	6.4590	0.1690
GRU	None	0.3641	0.0086	6.6024	0.1553
	Bus stops	0.3638	0.0098	6.5961	0.1783
	Tram stops	0.3706	0.0209	6.7202	0.3787
	Traffic signals	0.3631	0.0119	6.5842	0.2160
	Trees	0.3679	0.0126	6.6708	0.2277
	Traffic density	0.3637	0.0099	6.5948	0.1792
	Sidewalks	0.3641	0.0149	6.6019	0.2698
	All	0.3678	0.0154	6.6690	0.2793
TCN	None	0.3569	0.0045	6.4713	0.0821
	Bus stops	0.3580	0.0039	6.4909	0.0702
	Tram stops	0.3581	0.0064	6.4935	0.1158
	Traffic signals	0.3592	0.0071	6.5122	0.1281
	Trees	0.3568	0.0047	6.4686	0.0856
	Traffic density	0.3584	0.0052	6.4990	0.0935
	Sidewalks	0.3564	0.0039	6.4616	0.0709
	All	0.3547	0.0062	6.4317	0.1125
RFR 1	None	0.3258	0.0021	5.9072	0.0373
	Bus stops	0.3261	0.0017	5.9125	0.0310
	Tram stops	0.3262	0.0017	5.9140	0.0310
	Traffic signals	0.3266	0.0021	5.9219	0.0389
	Trees	0.3256	0.0027	5.9031	0.0489
	Traffic density	0.3260	0.0026	5.9106	0.0470
	Sidewalks	0.3257	0.0022	5.9057	0.0400
	All	0.3249	0.0023	5.8908	0.0423
RFR 2	None	0.3249	0.0017	5.8902	0.0309
	Bus stops	0.3249	0.0015	5.8901	0.0275
	Tram stops	0.3244	0.0017	5.8824	0.0314
	Traffic signals	0.3248	0.0010	5.8890	0.0181
	Trees	0.3248	0.0016	5.8883	0.0297
	Traffic density	0.3239	0.0018	5.8730	0.0321
	Sidewalks	0.3257	0.0013	5.9059	0.0229
	All	0.3241	0.0020	5.8757	0.0371

Table A.1: Mean and standard deviation values for each model and each OSM feature.