Udacity Machine Learning Nanodegree
Capstone Project Report

Predicting Swimming Pool Occupancy

Lukas Koucky

10 January 2020

1 Definition

1.1 Project Overview

I started swimming regularly in my local swimming pool Šutka swimming pool about two years ago. Usually when I'm leaving work I ask myself a question: "Should I go swimming today?" I can look at the web page of swimming pool to see how many visitors are in the pool right at that moment to get a hint if the swimming lines are reasonably occupied or overcrowded. But the problem is that it takes me another 45 minutes to get the pool during which the situation can change a lot. But what if I have a tool that will predict how many people will be in a pool when I get there or even tell me in the morning if today is a good day to visit the pool let's say between 5PM and 6PM? Goal of this capstone project was to develop such tool.

Predicting number of people in pool throughout the whole day is typical example of time series forecast. In this project more precisely multivariate time series prediction. In simple or univariate time series prediction models input is single series of observations and output is series of future predictions (i.e. number of visitors in the pool). On the other hand multivariate time series prediction produces the same output as univariate one but each input timestamp have multiple observations or features. In this project not just attendance data are the input for prediction algorithm but also number of reserved swimming lines, weather or public holidays at each time step. These problems are studied in both academic and professional sphere for example for financial market predictions [1], attendance prediction [2], power consumption prediction [3] or air pollution prediction [4] and many many more.

[1] https://arxiv.org/abs/1909.12789

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6109553/

[3] https://machinelearningmastery.com/household-power-consumption

[4] https://www.kaggle.com/c/dsg-hackathon

1.2 Problem Statement

There were two main goals for this project. First one was to train and deploy machine learning system that will be able to predict number of visitor in Šutka swimming pool throughout the day and the second was to create web page that can visualise predicted occupancy (number of visitors) at any time of the day using the trained model. To fulfill both tasks following was done:

- Gather all relevant data for machine learning algorithms
  - Historical data with number of pool visitors throughout the day are the most important data source for this task. But there are many more data inputs apart from historical data of swimming pool attendance that could be relevant for training of machine learning algorithms. For example weather or number of reserved swimming pool lines. First task was to discover all possibly useful data sources and gather all data.
- Clean and preprocess data into useful format
  - There are usually some errors in gathered data. Sometimes the webserver collecting occupancy data experience problems, sometimes webpages of pool are not accessible and these data points must be removed. There are also other possible situations where input data are not suitable for machine learning - for example days when pool was closed due to cleaning or holidays and attendance was therefore zero.
  - Other preprocessing steps like normalization, one hot encoding of some features and generating new features
  - Export to CSV and pickle format for easier processing
- Train 3 different machine learning algorithms for pool occupancy prediction on clean data
  - Several machine learning algorithms was inspected and trained for the task of prediction pool occupancy.
  - Following algorithms were inspected: Monthly average, Gaussian Mixture Model, Hidden Markov Model, Long Short Term Memory, Convolutional Neural Networks, Support Vector Machine, AdaBoost, Random Forest Classifier and Regressor, Extra Trees Classifier and Regressor
  - For supervised learning algorithms was time series transformed using sliding windows approach described in [2].
- Create web page that will visualize predictions of machine learning algorithms and inform user when is the good time to visit the pool for selected day

– Models trained in previous step were deployed and used for actual prediction on a web page that visualize prediction for selected day and offers possibility to compare results of multiple algorithms

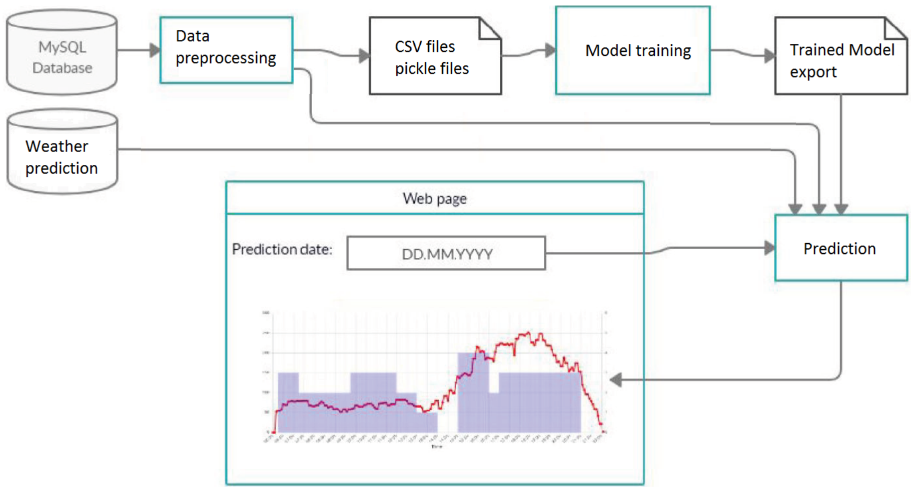

Figure 1: Diagram of project pipeline

1.3 Metrics

Mean squared error is used to measure performance of prediction. Prediction output is vector of integers representing number of people in pool. Error metrics for task like this is usually either Mean Absolute Error (MAE) or Mean Square Error (MSE). Both metrics computes average of differences between ground truth values and predicted values. The main distinction between these two metrics is that MSE uses square of difference while MAE uses absotulute difference. I chose MSE as final metrics because it punishes outliers more than MAE since the error grows quadratically with distance and this forces algorithm to minimalize number of outliers. Predictions are always generated for the whole day so it makes sense to measure performance also on the whole day.

$$MSE_{day}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \tag{1}$$

Mean square error of one day's prediction is computed using Equation 1 above where y is vector of ground truth data (real attendance during the whole day with 5 minute sampling). $\hat{y}$ is vector of predicted attendance in the same time steps as y. n is number of time steps for the whole day (usually 288). y_i is the i-th sample in vector y.

The performance of implemented algorithms will be measured as a mean of MSE_{day} on all days in testing dataset. This metric will show which algorithm performs better overall.

$$MSE_{dataset} = \frac{1}{k} \sum_{day=0}^{k-1} MSE_{day}(y, \hat{y}) \tag{2}$$

Where k is the number of days in dataset. y is vector of ground truth data for given day and $\hat{y}$ is vector of predicted attendance for given day.

TODOs:

- Promazat jake algoritmi byly prozkoumane aby to odpovidalo v Train 3 different machine learning algorithms...
- Final read if it makes sence + check answers to questions in template.

2 Analysis

2.1 Data Exploration

Every machine learning project starts with the data gathering and analysis. Data for this project comes from several sources. The most important data input is collected on my personal webserver that stores pool occupancy every 5 minutes into the MySQL database from publicly available data on Šutka pool web page. Data are collected for past 2 years which means I have more than enough data for training, validation and testing of machine learning algorithms. But there are also many other inputs that influence attendance of swimming pool and are highly valuable for predictions. One of the most important is number of reserved swimming lines with names of organisations reserving the lines. This information is also collected from publicly available data on Šutka pool web page. Next important input is time of the year that is included in timestamp of each timesample, public holidays acquired from officeholydas.com and finally local weather acquired from archive of in-pocasi.cz. Local weather contains information about temperature, humidity, precipitation, wind strength and air pressure from multiple stations across Prague. All these information are also stored in the MySQL database. See Image 2 for the sample export from database. History weather information is collected in database but for the occupancy prediction into the future is used in-pocasi.cz free predictions REST API. Working directly with MySQL database would not be very comfortable. This is why the database was exported, preprocessed and saved to csv files and pickle files. More about preprocessing can be found in section 3.1.

2.1.1 Pool attendance data

Attendance data are stored in table *occupancy*. Data are collected every 5 minutes which results in 288 time samples from each day. As you can see on mysql> select * from lines_usage limit 10;

id	date	time_slot	reservation
1072	2019-01-05	4	Neptun,
1073	2019-01-05	5	Neptun,
1074	2019-01-05	6	Neptun,
1075	2019-01-05	7	Neptun,
1076	2019-01-05	8	Neptun,
1077	2019-01-05	9	Neptun,
1078	2019-01-05	10	Neptun,
1079	2019-01-05	11	Neptun,
1080	2019-01-05	12	Neptun,
1081	2019-01-05	13	Neptun,

mysql> select * from occupancy limit 10;

id	percent	pool	park	lines_reserved	time	day_of_week
1	85	246	180	0	2017-10-14 15:59:06	5
2	84	229	191	0	2017-10-14 16:05:06	5
3	83	236	179	0	2017-10-14 16:15:29	5
4	83	236	179	0	2017-10-14 16:16:30	5
5	83	236	179	0	2017-10-14 16:17:00	5
6	83	236	179	0	2017-10-14 16:18:00	5
7	83	236	179	0	2017-10-14 16:19:01	5
8	86	256	173	0	2017-10-14 16:20:01	5
9	86	256	173	0	2017-10-14 16:21:02	5
10	86	256	173	0	2017-10-14 16:22:06	5

mysql> select * from weather_history limit 10;

id	time	temperature	wind	humidity	precipitation	pressure	station
1	2018-11-17 00:30:00	2.6	7	85	0.0	1034.7	1
2	2018-11-17 01:00:00	2.5	5	85	0.0	1034.8	1
3	2018-11-17 01:25:00	2.4	4	85	0.0	1034.7	1
4	2018-11-17 01:55:00	2.3	4	86	0.0	1034.9	1
5	2018-11-17 02:26:00	2.1	5	86	0.0	1035.0	1
6	2018-11-17 02:55:00	1.9	5	86	0.0	1034.8	1
7	2018-11-17 03:27:00	1.9	4	87	0.0	1035.0	1
8	2018-11-17 03:55:00	1.6	7	87	0.0	1034.7	1
9	2018-11-17 04:26:00	1.4	7	88	0.0	1034.9	1
10	2018-11-17 04:55:00	1.4	6	88	0.0	1034.9	1

Figure 2: Sample export of 10 rows from all tables in MySQL database. Top left table *lines_usage* with information about which organisation reserved line in given time. Top right table *occupancy* with information about number of people in the pool. Bottom table *weather_history* with information about weather history in Prague.

Image 2 top right there are more than just pool attendance stored for each time stamp. Following information are store in this table:

- **id** - unique identifier integer of data sample.
- **percent** - occupancy of pool in percentage. 100 means that pool is full.
- **pool** - number of people in the pool area.
- **park** - number of people in the park area.
- **lines_reserved** - not used any more, set to 0. Number of reserved lines together with additional information are stored in table lines_usage described in section 2.1.2.
- **time** - timestamp with date and time.
- **day_of_week** - day of week starting with 0 for Monday.

Notice that there are number of people in pool and in park. This is because the whole Šutka pool is divided into two parts. First part is pool with 50 meters swimming pool, small pool for little children, showers, two steam rooms, two saunas and resting area. Second part is park with small water park (water slides, whirlpools, outdoor relax area, foot court and bar). These two parts of the pool are connected with tourniquets so the number of people in each area is precisely monitored. Since only prediction of number of people in pool is the scope of this project columns *percent* and *park* are not used. Also *lines_reserved* is not used since this information is stored in separate table *lines_reserved*.

2.1.2 Lines usage data

Table *lines_reserved* stores information about reserved lines. Reservations can be made for 15 minutes time slots and each organisation can reserve more than one line for any number of time slots. Therefor names of the organisations and number of lines in each time slot are stored. If there is no entry for particular day or time it means that no line is reserved. Since only time slot and names of organisations that rents line in particular time slot is saved there is no information about which lines precisely are booked - only how many lines are booked. But it is not necessary to know which specific line was reserved to predict attendance, information about organisation and number of lines is sufficient. Description of table columns is following:

- **id** - unique identifier integer of data sample
- **date** - date in format YYYY-MM-DD
- **time_slot** - integer from 0 to 63 that represents time slot of swimming line reservation in reservation table on Šutka swimming pool web page
- **reservation** - comma separated strings containing names of clubs or organisations that rented the line. One name can be represented multiple times which means that given organisation rented multiple swimming lines

2.1.3 Weather data

Table *weather_history* stores historical data of weather in Prague. Data are collected from in-pocasi.cz archive.There are several measurement stations available each with the unique id in column station. Description of table columns is following:

- **id** - unique identifier integer of data sample
- **time** - timestamp with date and time
- **temperature** - temperature in degrees of Celsius
- **wind** - wind strength in meters per second
- **humidity** - humidity in percents
- **precipitation** - rain or snow precipitation in millimeters
- **pressure** - air pressure in kPa
- **station** - unique id of weather station where the measurement comes from

	pool	lines_reserved	holiday	temperature_binned	wind_binned	humidity_binned	precipitation_binned	pressure_binned
count	129440.000000	129440.000000	129440.000000	129440.00000	129440.000000	129440.000000	129440.000000	129440.000000
mean	105.066061	1.181768	0.006528	3.82021	1.391656	2.786295	0.175549	1.924992
std	74.830401	1.309986	0.080533	1.78597	1.021348	0.912442	0.498330	0.905747
min	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000
25%	50.000000	0.000000	0.000000	2.00000	1.000000	2.000000	0.000000	1.000000
50%	88.000000	1.000000	0.000000	4.00000	1.000000	3.000000	0.000000	2.000000
75%	163.000000	2.000000	0.000000	5.00000	2.000000	3.000000	0.000000	2.000000
max	378.000000	8.000000	1.000000	7.00000	5.000000	4.000000	3.000000	4.000000

Figure 3: Statistics of selected data.

2.1.4 Data statistics

Image 3 shows statistics of selected data after preprocessing. It is important to note that this means that all the data from hours when pool is closed were not used. This resulted in 129440 time samples. There are all available data except time stamps and also all line reservation information are represented in column *lines_reserved*. Column *pool* represents attendance of pool and column *holiday* represents public holidays in binary form (1 - day was public holiday, 0 - day was not public holiday). Remaining columns ending with *_binned* are weather data after preprocessing described in section 3.1. From the data is visible that average attendance throughout the whole dataset at any time is 105 people with maximum attendance 378 people. There are on average 1.18 reserved lines with almost no precipitation and average temperature around bin number 4 which is from 10 to 15 degrees Celsius (more about weather data binning in section 3.3.1). These statistic data didn't produce as much information as visual exploratory of data in next section.

TODOs:

- If a dataset is present for this problem, have you thoroughly discussed certain features about the dataset? Has a data sample been provided to the reader?
- If a dataset is present for this problem, are statistics about the dataset calculated and reported? Have any relevant results from this calculation been discussed?
  - Add statistics - average attendance, average lines reserved at each time (image), number of time samples, number of organisations, some interesting graphs of day vs feature?
- Are there any abnormalities or characteristics about the input space or dataset that need to be addressed? (categorical variables, missing values, outliers, etc.)

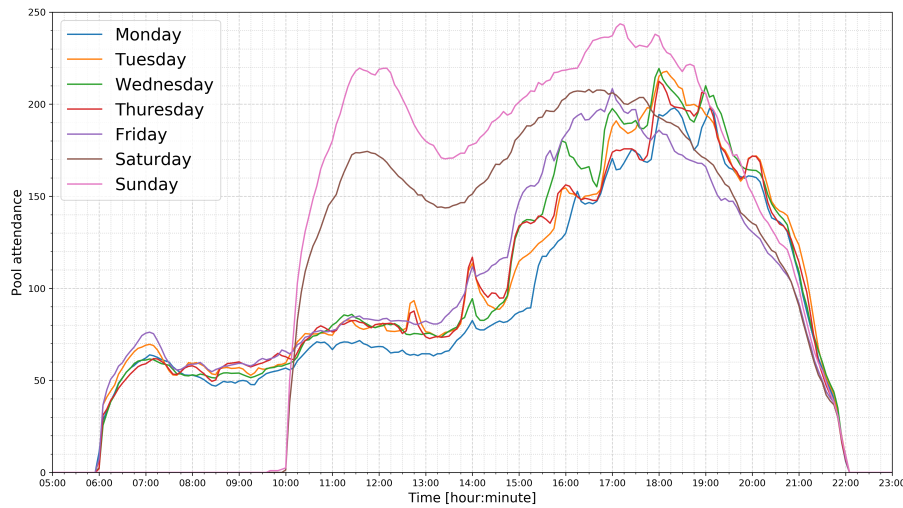

Figure 4: Average attendance for each day of week

2.2 Exploratory Visualization

Image 4 shows average attendance at each day of week. This image shows several very important patterns in data. Most obvious is difference between weekend and weekday. On weekdays is pool open from 6:00 and attendance quickly rise to 50 people. Than from around 14:00 attendance rise again until 18:00 when it starts to drop until closing time at 22:00. On weekend days pool opens at 10:00 which is followed by steep increase of occupancy that drops a little around 12:00, then rise again and from 18:00 decrease until closing time at 22:00.

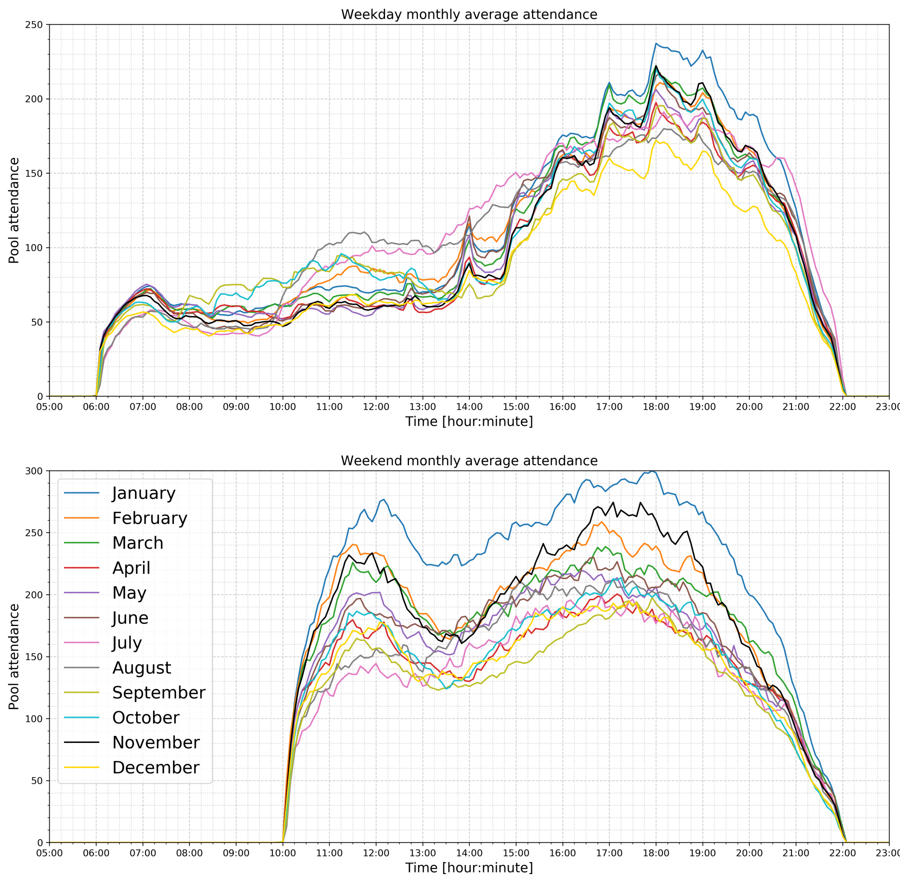

Figure 5: Monthly average attendance. Top image shows average attendance during weekdays. Bottom image shows average attendance during weekend days.

Image 5 shows average attendance at each month at weekdays (top) and weekends (bottom). This Image shows another important pattern in data and it is the seasonality. It is interesting that weekend attendance is much more influence by seasonality than weekdays. The difference between average attendance at 18:00 in January (the busiest month) and December (the calmest month) is 70 people on weekdays but 120 on weekends. December is also interesting from another perspective. The average attendance rises from July to January with the only exception in December. This could be caused by Christmas holidays and New Year when many people have vacations, spending time with family and friend and there are also many public holidays.

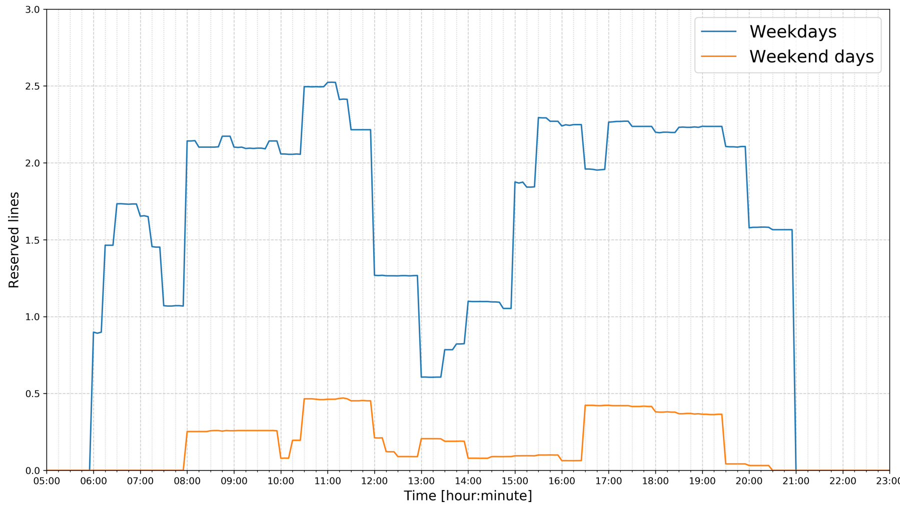

Figure 6: Average number of reserved lines on weekdays and weekend days.

Image 6 shows average number of reserved lines on weekdays and weekend days. Weekends are usually without any reservations, that is also why average number reserved lines is always under 0.5. Occupation on weekdays on the other hand are highly influence by reserved lines. There is clear pattern with over 2 lines reserved on average from 8 to 12 o'clock and than the similar pattern between 15 and 20 o'clock. This is also why I think that information about lines reservation will be important features for machine learning algorithms.

TODOs:

- Have you visualized a relevant characteristic or feature about the dataset or input data?
- Is the visualization thoroughly analysed and discussed?
- If a plot is provided, are the axes, title, and datum clearly defined?
- Image - Histogram of how often the organization is present
- Image - Interesting organisation and why to use them
- Image - Unusual data. Are there any?

2.3 Algorithms and Techniques

TODOs:

- Are the algorithms you will use, including any default variables/parameters in the project clearly defined?

- Are the techniques to be used thoroughly discussed and justified?
- Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?

2.3.1 Random Forest Classifier and Regressor

Random Forest is an ensemble method that combines multiple decision trees. Each decision tree produces prediction and the most frequent prediction is chosen as the output of Random Forest Classifier or average prediction for Random Forest Regressor. Randomness in name comes from random bagging during learning phase. Each decision tree is learned on different subset of training data generated by randomly sampling the training set. Also the decision at each node is made on random sub-sample of available features and the best split is used [5] [3].

2.3.2 Extra Tree Classifier and Regressor

Extra Tree is an ensemble learning method utilizing decision trees similar to Random Forest. There are however few differences between these two classifiers. Its name is coming from extremely randomized trees and was proposed by [4]. The main difference from Random Forest is that Extra Tree does not bootstrap observations and splits node randomly. This means that randomness does not come from random data bootstrapping but from random splits of observations [1].

2.3.3 CNN

2.3.4 LSTM

2.4 Benchmark

TODOs:

- Has some result or value been provided that acts as a benchmark for measuring performance?
- Is it clear how this result or value was obtained (whether by data or by hypothesis)?

Just by looking at the Image 4 we can clearly see that average attendance progress throughout all weekdays looks similar. Same is true for weekend days. On the other hand looking at Image 5 we can definitely see seasonality trends in each month. For example January afternoon attendance is almost 100 more than September attendance. This is why I chose monthly average (one for weekdays and one for weekends) to be the benchmark prediction model. It provides reasonable good approximation of real attendance throughout the month since it usually does not change a lot during one month.

On the other hand, when comparing each day individually we can see some minor differences during the day's progress (see Image **??**) that may be caused by weather, line reservations, public holidays or some other features. I hope to see that machine learning algorithms will be able to spot these minor differences and utilize these features to provide more accurate predictions than simple monthly average.

2.4.1 Monthly average

For each time stamp throughout the day is computed average attendance at given month. Since the attendance on weekdays and weekends differs highly (see Image 4) two average predictions are generated for each month - one for weekdays and one for weekends.

$$\hat{y}_i = \frac{1}{m} \sum_{j=0}^{m-1} y_j \tag{3}$$

Where $\hat{y}_i$ is prediction at time stamp i. m is number of time samples in training set for predicted month in given week time (weekday or weekend).

3 Methodology

TODOs:

- Implement your algorithms and metrics of choice, documenting the preprocessing, refinement, and postprocessing steps along the way.
- Rozdeleni na tridy mozna class diagram
- Preprocessing: Mysql -¿ sqlite -¿ csv -¿ pickle of days
- Preprocessing removeing of bad days (all zeros, too short, dont start from begining of day, closed days, reserved competition)
- Algorithms used from sklearn, keras and HMM
- Mean squered error implementation by day - using old prediction for new time steps
- Own grid tuning implementation and hyperparameter tuning
- I had to tune params of algorithm + what features to use + time step back
- Web page implementation - js knihovna + date picker + image of how it looks and show different possibilities
- Prediction done by cron running on webserver every day

Implementation can be split into several steps that leads to complete web page with working predictions. This section is split into several parts based on these steps. They are:

- Data preprocessing
- Algorithms implementation and tuning
- Postprocessing
- Web page implementation

3.1 Data Preprocessing

Origin of data and it's structure in MySQL database is described in previous section 2.1. In this section are more in depth discussed preprocessing steps to make data easily accessible for machine learning algorithms fitting and predictions.

First part of preprocessing was to convert data from MySQL database to csv file. Database contains three tables with information about attendance, swimming pool lines reservation and weather. As mentioned in 1.2 public holidays should be also used but are not present in database. Public holiday dates for all years in dataset were downloaded from officeholydas.com. For the export I decided to use structure of table *occupancy* where each row represents one time stamp every 5 minutes. To each time stamp is than exported number of reserved lines and which organisations are reserving lines, all weather information from table *weather_history* at measurement station closest to the pool and flag if this day is public holiday or not. Result of this preprocessing using Pandas is Data Frame with following structure:

	pool	lines_reserved	time	day_of_week	month	day	hour	minute	holiday	reserved_Lavoda	...	reserved_OS DUFA	temperature_binned	wind_binned	humidity_binned	precipitation_binned	pressure_binned	reserved_other	minute_of_day	year
29812	0	0	2018-03-13 04:00:11	1	3	13	4	0	0	0	...	0	3	2	4	0	0	0	240	2018
29813	0	0	2018-03-13 04:05:03	1	3	13	4	5	0	0	...	0	3	2	4	0	0	0	245	2018
29814	0	0	2018-03-13 04:10:09	1	3	13	4	10	0	0	...	0	3	2	4	0	0	0	250	2018
29815	0	0	2018-03-13 04:15:05	1	3	13	4	15	0	0	...	0	3	2	4	0	0	0	255	2018
29816	0	0	2018-03-13 04:20:06	1	3	13	4	20	0	0	...	0	3	2	4	0	0	0	260	2018

Figure 7: Structure of Data Frame after initial preprocessing and refinement that connects all available data into one table.

You can see on Image 7 that only information used from *occupancy* table is time stamp and pool attendance. Other data are not relevant for the occupancy of swimming pool itself. Another preprocessing step visible on Image 7 is reshaping of organisations reservation data. Database table *lines_usage* store names of organisation reserving lines in comma separated string while in this

Data Frame have each organisation it's own column and value on each row represents number of lines reserved by given organisation in time stamp at that row.

Since all the algorithm used utilizes supervised learning a sliding windows approach described in [2] was used to serve data. At each time stamp are all available features provided. This creates vector of input features with following content:

$$v(t) = (attendance(t), lines(t), minute(t), day, month, year, weather(t), org(t)) \tag{4}$$

Where

- t is the time step of the day. Each day is sampled with 5 minute steps. So for example $t = 0$ is time 0:00, $t = 3$ is time 0:15 and $t = 287$ is time 23:55,
- $v(t)$ is vector of features at time t.
- $attendance(t)$ is pool attendance at time t.
- $lines(t)$ is number of reserved lines at time t.
- $minute(t)$ is minute of the day at time t. Minute of the day is counted from the beginning of the day.
- day is the day of week indexing from 0 as Monday. So for example 1 is Tuesday and 6 is Sunday.
- $month$ is the month of the year indexing from 0 as January. So for example 3 is April and 11 is December.
- $year$ is current year with offset 2015. So 4 is year 2019.
- $weather(t)$ is vector of weather data at time t. Weather data contains following in this particular order: temperature, wind, humidity, precipitation, pressure. All the weather values are binned as described in section 3.1.
- $org(t)$ is vector of organisations reserving swimming lines at time t. Precise content of this vector is described in section 3.1.

TODOs:

- How DB looks - tables, image of tables
- Convert to CSV
- split of data to training testing and validation
- Day class
- pickle of days

3.2 Implementation

3.2.1 Data preprocessing pipeline

3.2.2 Helper classes

Utils (Days class, feature producers), DataHelper (days prediction, graphs), DaysStatistics

3.2.3 Monthly average

3.2.4 Random Forest Classifier and Regressor

3.2.5 Extra Tree Classifier and Regressor

3.2.6 Convolutional Neural Network

Convolution Neural Network (CNN) is a type of deep learning method utilizing convolution mathematical operations.

Throughout the implementation were tested many different models with many parameter settings. The most It is most commonly used f

TODOs:

- More about CNN
- How is it used - cite brownlee2019cnn
- problems and iterations of different CNNs

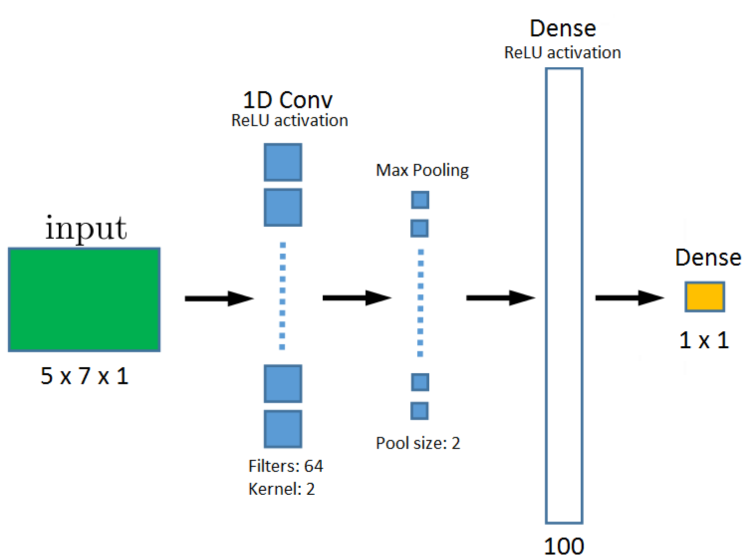

Figure 8: Convolutional Neural Network architecture

3.2.7 Long Short Term Memory

TODOs:

- More about LSTM
- How is it used - cite brownlee2019lstm
- problems and iterations of different LSTM architectures, mention bidi-rectinal

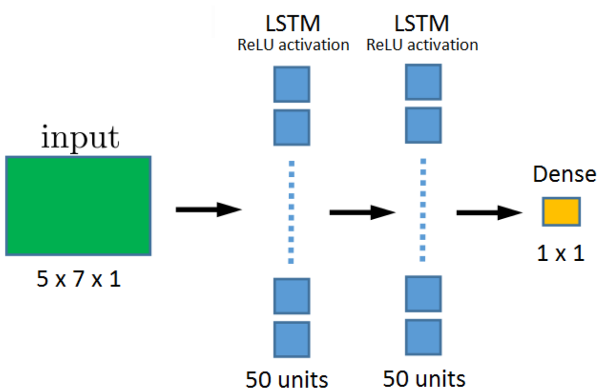

Figure 9: Long Short Term Memory Network architecture

3.2.8 Notebooks

3.2.9 Web application

3.3 Refinement

3.3.1 Bin weather data

Weather data presented large spectre of values but initial tests on Random Forests and LSTMs shows that this data had very little information gain and increase size of Random Forrest and training time of LSTMs. That is why I decide to bin weather data to followong bins:

- temperature - temperature in degrees of Celsius to following 8 bins: $(-100, -5], (-5, 0], (0, 5], (5, 10], (10, 1$
- wind - strength in meters per second to following 6 bins: $(-1, 1], (1, 5], (5, 10], (10, 15], (15, 20], (20, 1000]$
- humidity - humidity in percents to following 5 bins: $(-1, 20], (20, 40], (40, 60], (60, 80], (80, 100]$
- precipitation - rain or snow precipitation in millimeters to following 4 bins: $(-1.0, 0.1], (0.1, 5.0], (5.0, 10.0], (10.0, 1000.0]$
- pressure - air pressure in kPa to following 5 bins: $(0, 1000], (1000, 1010], (1010, 1020], (1020, 1030], (1030, 20$

3.3.2 Remove organisations with few reservations

Encoding each organisation as new column in Data Frame with all features generated more than 100 extra columns for more than 100 organisations. But many organisations had just very few reservations and would not be beneficial to keep this information in Data Frame. That is why only organisations that regulary reserve lines and therefor their reservation could have significiant influence of overall attendance were kept in Data Frame. All the other organisations (with less than 200 reservation in total) were compressed into one column *reserved_other*. This led to reduction of number of columns to reasonable XX TODO ADD CORRECT NUMBER.

3.4 Limit output of prediction to 0 - 400

3.5 Implementation of my own grid search

Regualr grid search is using only MSE of one time sample prediction. This means all algorithms are quickly close to zero error. Explain that I implemented grid seach that compares outputs of prediction on whole day prediction where output of prediction in one time step is input to prediction in next step.

4 Results

4.1 Model Evaluation and Validation

TODOs:

- Collect results about the performance of the models used, visualize significant quantities, and validate/justify these values.
- Base results on monthly average
- comparisn to of monthly average to other algos
- influence of parameter tuning on algorithms
- image with MSE of all methods on testing data - bar chart
- image for RFC and ETC with n_estimators on x axis and MSE on y axis

4.2 Justification

TODOs:

- Are the final results found stronger than the benchmark result reported earlier?
- Have you thoroughly analysed and discussed the final solution?
- Is the final solution significant enough to have solved the problem?

5 Conclusion

5.1 Free-Form Visualization

TODOs:

- Construct conclusion about your results, and discuss whether your implementation adequately solves the problem.
- What was good and what bad
- Further improvements

5.2 Reflection

5.3 Improvement

References

[1] Naman Bhandari. Extra trees classifier. https://medium.com/@namanbhandari/extratreesclassifier-8e7fc0502c7, 2019. [Online; accessed 12-December-2019].

[2] Jason Brownlee. How to convert a time series to a supervised learning problem in python. https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/, 2019. [Online; accessed 10-November-2019].

[3] Wikipedia contributors. Random forest — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Random_forest, 2019. [Online; accessed 12-December-2019].

[4] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Mach. Learn.*, 63(1):3–42, April 2006.

[5] Tony Yiu. Understanding Random Forest. https://towardsdatascience.com/understanding-random-forest-58381e0602d2, 2019. [Online; accessed 12-December-2019].