

DLMDSME01

Automation of Standby Duty Planning for Rescue Drivers via a Forecasting Model

GitHub: XXXXXXXXXXXXXXXX

Niels Humbeck

32003045

19.11.2021

Köln, Germany

# Table of contents

[1 Table of contents 2](#_Toc88930736)

[2 Image directory 3](#_Toc88930737)

[3 Abbreviations 3](#_Toc88930738)

[4 Introduction: 4](#_Toc88930739)

[5 Business understanding 5](#_Toc88930740)

[6 Data Acquisition & Understanding 6](#_Toc88930741)

[6.1 Datatype, missing values and outliers 6](#_Toc88930742)

[6.2 Data wrangling & analysis 7](#_Toc88930743)

[6.3 Calculating the base line success KPIs 9](#_Toc88930744)

[7 Model Training and Evaluation 10](#_Toc88930745)

[7.1 Model requirements and library selection 10](#_Toc88930746)

[7.2 Model Training and evaluation 11](#_Toc88930747)

[8 Deployment 11](#_Toc88930748)

[9 Conclusion 11](#_Toc88930749)

[10 Library 12](#_Toc88930750)

# Image directory

# Abbreviations

# Introduction:

Model Engineering … very important --- case study to get experience

XX% of the data science project make it to the development. The low success rate has different reasons. One reason is the low description of the use case another reason is the structure of data science projects. In the given work we describe based on the Teams Data Science Process methodology a use case of model engineering for a automation of standby duty planning for rescue drivers.

In the first chapter the used methodology for the given case study in model engineering is explained. The applied framework is called “Teams Data Science Process (TDSP)”. TDSP is a framework providing a structural methodology to conduct data science projects. Based on the framework the first chapter provides basic understanding of the business. The focus is set here especially on the problem description, the goal of the data science project, the measures how to quantify the success of the project and the benefits. The third chapter is about data acquisition & understanding. In the fou

The furth chapter….

Teams Data Science Process (TDSP)

The core concept of the TDSP is depicted in the figure below (see Figure 1).

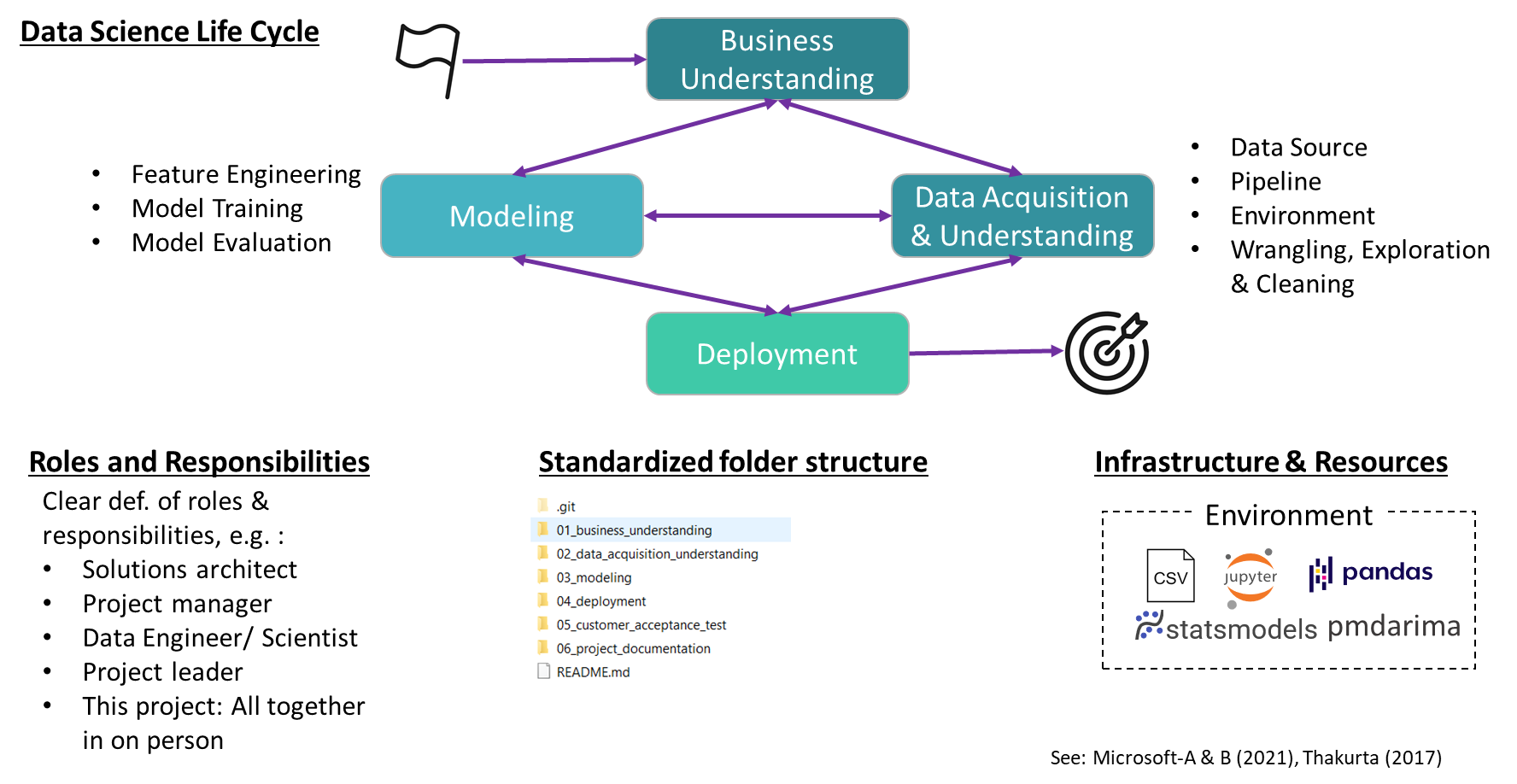


Figure 1: The Teams Data Science Process (TSDP)

The TDSP contains a data science life cycle, the standardized roles and responsibilities, the standardized folder structure and the infrastructure & resources (Microsoft-A, 2021). The data science life cycle described in s flexible and iterative manner a methodic ho to conduct data science project. The very first step is to understand the business and its needs. Together with the subject matter experts – who has the needed domain knowledge – the data scientist describes the problem and outlines the goal of the data science projects. It is important to define in the beginning quantifiable success measures. In the next step the data is acquired and analyzed: Which data sources are available, creating data pipelines, accessing the quality of the data, cleaning the data, exploring hidden patterns in the data. Based on the given data a model is trained in order to serve the business needs. Feature engineering is conducted before model training to provide meaningful input values. After the model is trained a evaluation is conducted. If the evaluation meet the measurable success criteria, the model is deployed in production. All of this steps are iterative an interconnected. For example a first trained model is shown to the subject matter expert to get a feedback (Microsoft-B, 2021).

The TDSP outlines the key personal roles and associated tasks. Basic roles in a data science project are: solutions architect, project manager, data engineer, data scientist, project lead (Thakurta, 2017). Besides roles and responsibilities the framework advocates clear folder structure as well as the use of a version control software to enable the team work. In the given project git hub is used as the version control system. The used folder structure is depicted in Figure 1. Furthermore TDSP gives advises for the infrastructure to use in a data science project.

# Business understanding

Berlins red-cross is a charity-oriented organization providing ambulance transport services. The organization incorporates 51000 members, 2500 volunteer workers and 1000 full time employees. Operating ambulance transports is a ethical sensible environment since lives depends on the reliability and availability of these transports. That’s why sufficient capacities of ambulance transport is eminent to the business success.

The business faced in the past difficulties with insufficient planning of the standby-duty planning of rescue drivers. The number of rescue drivers needed highly depends on the amount of emergency calls received per day. For each day a predefined number of rescue driver as well as standby rescue drivers is on duty. Short-term sickness of rescue drivers as well as unusual high amounts of emergency calls results in an unusual high demand of rescue drivers which can exceed the amount of available rescue drivers. Unusual low short-term sickness of rescue drivers as well as unusual low amounts of emergency calls – on the other side - results in an unusual low demand of rescue drivers. In this case the amount of planned standby drivers are not needed. The goal of the given data science project is to create a model which predicts on the 15th for the upcoming month the demand of rescue drivers (inc. Standby divers) for the next month. The prediction is influenced by seasonal patterns. The success of developed model is measured by:

* percentage of standbys being activated is higher than in the current approach of keeping 90 drivers on hold
* situations with not enough standbys should occur less often than in the current approach.

A successfully deployed prediction model in the production has several benefits:

* Improved prediction of demand of rescue drivers results in less cost for providing idle capacities
* Higher reliability in duty planning returns in higher free time quality of employees
* Increase in trust in the capabilities of Berliner Red-Cross to cope with the

# Data Acquisition & Understanding

## Datatype, missing values and outliers

The data is provided in a csv file and contains daily datasets starting from the 2016-14-01 till 2019-05-27. The dataset contains following columns:

* date: entry date [Datetime index]
* n\_sick: number of drivers called sick on duty [integer]
* calls: number of emergency calls [float]
* n\_duty: number of drivers on duty available [integer]
* n\_sby: number of standby resources available [integer]
* sby\_need: number of standbys, which are activated on a given day [float]
* dafted: number of additional drivers needed due to not enough standbys [float]

It seems reasonable, that the number of sick drivers (n\_sick), the drivers on duty (n\_duty) and the number of standby resources available (n\_sby) are integer values since there are no half drivers. The columns calls, standby drivers needed (sby\_need) as well as the column dafted (number of additional drivers needed due to not enough standbys) are float values. In this use case no float values are needed. The float values were transformed into int64 datatype.

The data was inspected for **missing data**. All datasets were valid and didn’t contain missing information. Thus no data imputation is needed. In the Figure 2 are the distribution of all variables shown in a boxplot chart. The variables calls, n\_sick, n\_duty contains no unreasonable **outliers**. The variable n\_sby is a fixed number.

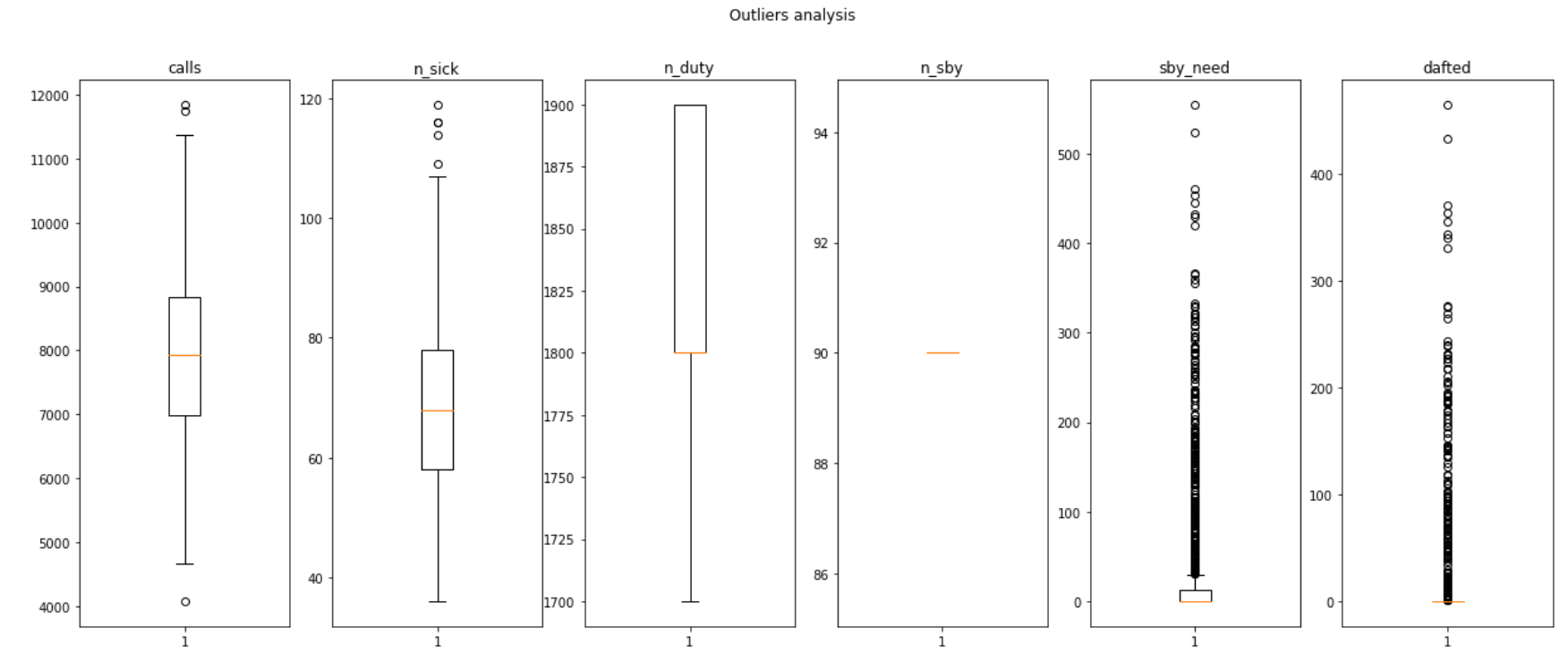


Figure 2: Boxplots for outlier analysis

The variables sby\_need and dafted show a high number of outliers. This is reasonable since sby\_need is in most cases zero unless standbys need to be activated. Same applies for the variable dafted when more than the duty + standby drivers are needed (see Figure 3 and Figure 4). All together no unreasonable outliers were detected.

|  |  |
| --- | --- |
| Figure 3: Histogram of standby drivers needed | Figure 4: Histogram of further drivers needed [exceeding duty and standby] |

## Data wrangling & analysis

For further analysis data wrangling is needed. Data can be reshaped and transformed to get valuable insights, e.g. with the Pandas function resample the average value of one month can be calculated. The Pandas DataFrame is indexed with the date column to facilitate further analysis. In some cases new columns were created by applying calculation, e.g. creating the column drivers\_atwork by adding the number drivers on duty (n\_duty) and the number of standbys needed (sby\_need).

The Figure 5 depicts the variables graphical over the given time period. The variable drivers\_atwork were created by adding n\_duty and sby\_need. The variables calls and n\_sick seems to have a seasonal as well as a trend component. The trend is positive which means over the time more calls and more drivers are sick. The variable n\_duty is jump-fixed with a trend upwards. The variable drivers\_atwork has as a lower boundary n\_duty. Beside this base line there are days where more driver needs to be at work. There seems to be a correlation between the number of calls and the days where more drivers are needed at work than the base line n\_duty and this pattern seems to have a seasonal and a trend component.

The variables sby\_need and dafted are in the most cases zero when the number of drivers on duty are sufficient. The relationship between between sby\_need & dafted can be explained as follows max(sby\_need-n\_sby;0). The pattern seems to be seasonal similar to the variable calls.

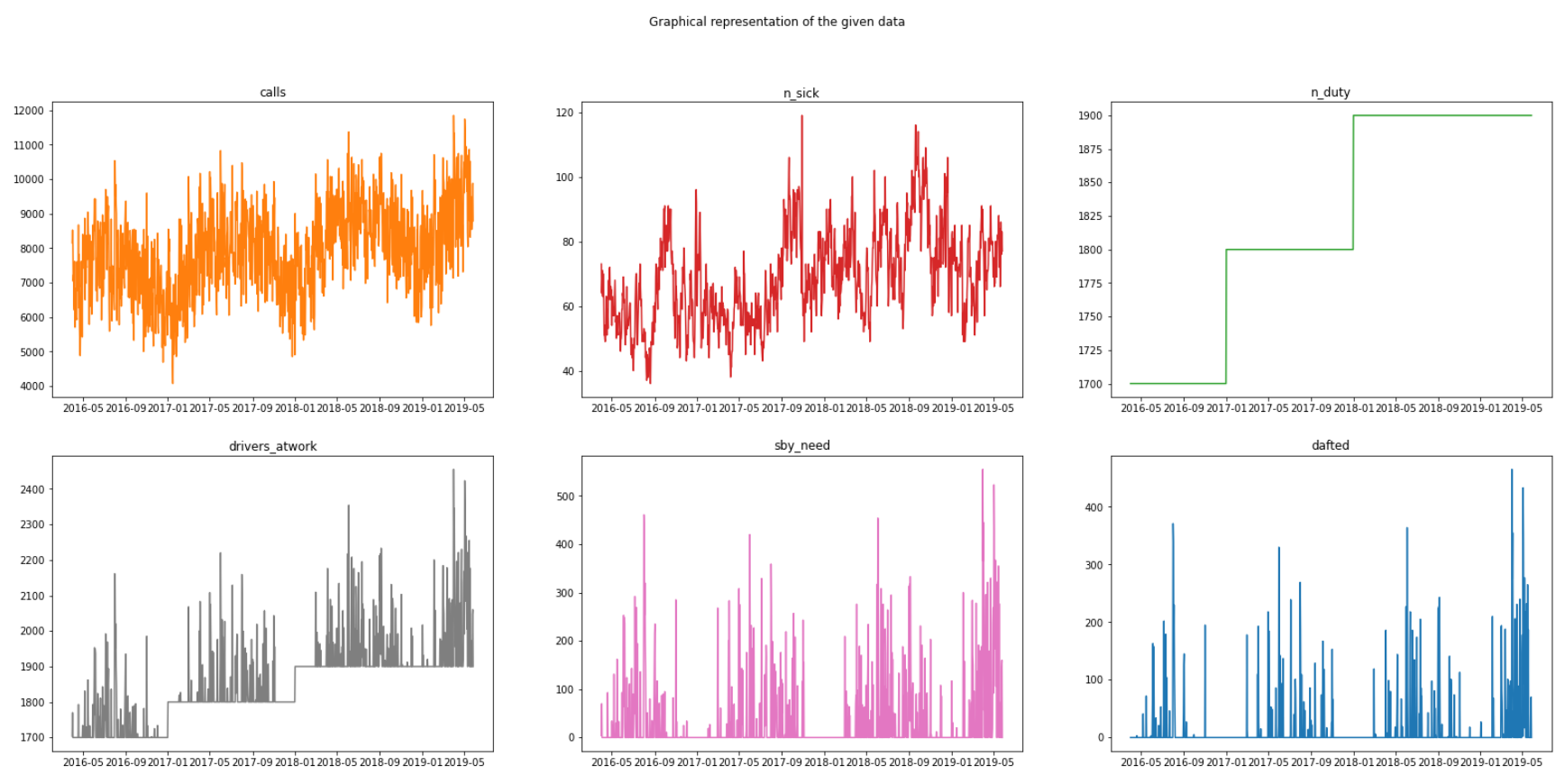


Figure 5: Visualization and data representation

The **seasonality** of the variables calls, n\_sick and drivers\_atwork were analyzed. There is a weekly trend for the variable calls as shown in Figure 6. This applies for n\_sick and drivers\_atwork as well.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Figure 6: Seasonality component shows a weekly seasonality of the variable calls

Besides the weekly seasonality exists a clear yearly seasonality for the variable calls and the variables driver\_atwork (see Figure 7 and Figure 9). The variable n\_sick seems to have a annual pattern as well where the most numbers of absenteeism occur in September and November (see Figure 8). The month plots were created by resampling the data.

|  |  |  |
| --- | --- | --- |
| Calls | N\_sick | Drivers\_at\_work |
| Ein Bild, das Antenne enthält.  Automatisch generierte Beschreibung  Figure 7: Month\_plot calls shows a seasonality over the year | Ein Bild, das Objekt, Antenne enthält.  Automatisch generierte Beschreibung  Figure 8: Month\_plot n\_sick shows a seasonality over the year | Ein Bild, das Objekt, Antenne enthält.  Automatisch generierte Beschreibung  Figure 9: Month\_plot drivers\_at\_work shows a seasonality over the year |

A **correlation analysis** has been conducted. The variables calls and drivers\_atwork are highly correlated (0,7 Pearson correlation). One distorting factor for the correlation analysis is the lower boundary of the number of drivers\_atwork (1700 , 1800, 1900). A reasonable guess is that less than the lower boundary of drivers\_atwork are needed for some periods. From a logical perspective there is a **causation** between the number of calls and the resulting number of drivers needed. An estimated guess is that there is a linear relationship between the amount calls and the resulting number of drivers\_atwork. Calculation of the linear coefficient describing best the relationship between calls and drivers at work: 4,82 calls per driver shift (with a reasonable MSE of 196). The result is shown in the Figure 10 (yellow calls/4,82 and black drivers\_atwork).

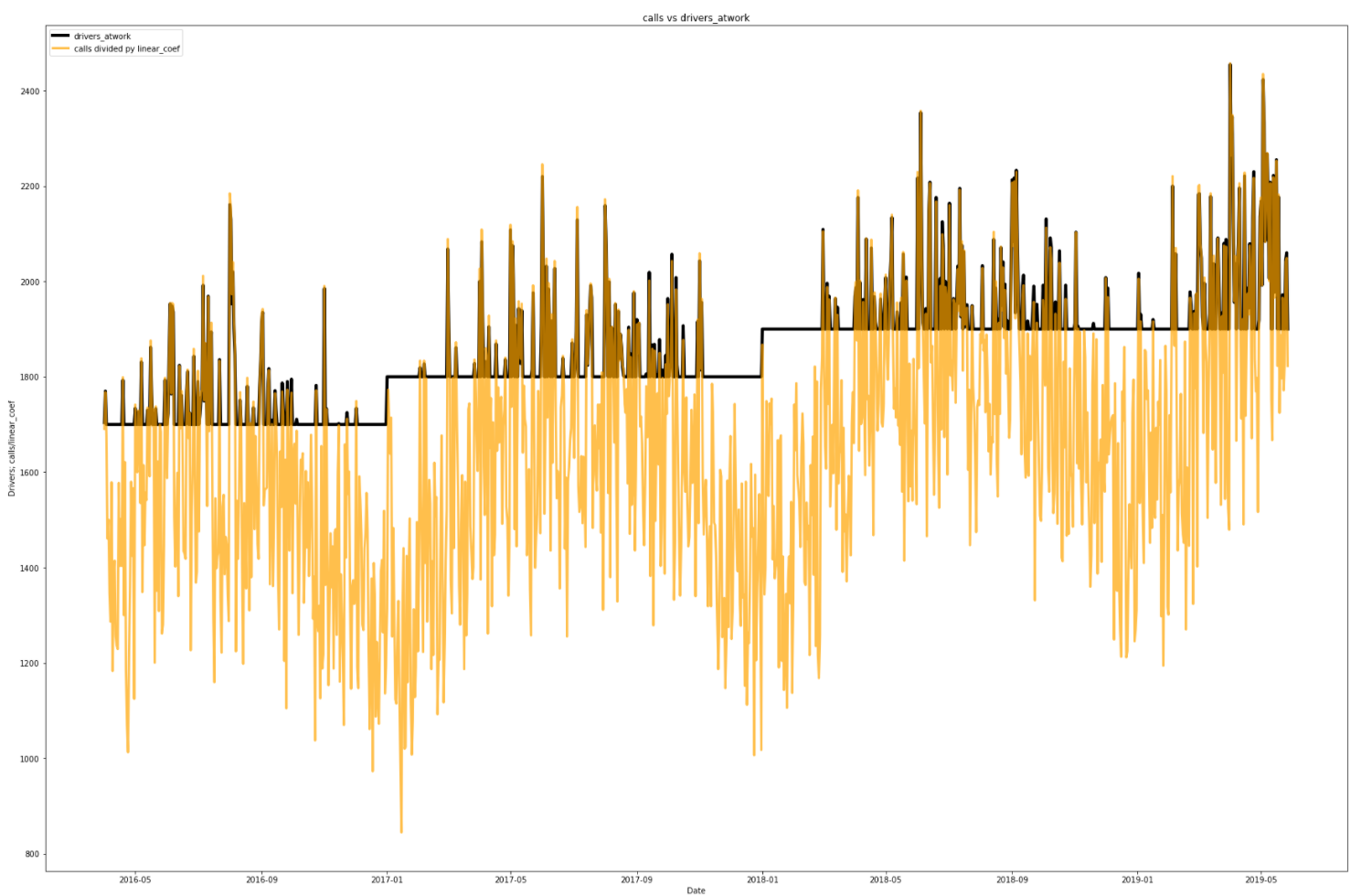


Figure 10: Linear correlation between calls and drivers\_atwork (linear coef. of 4,82 calls per driver shift)

## Calculating the base line success KPIs

In order to evaluate the prediction model KPI (Key Performance Indicators) needs to be defined and calculated for the current status. The business provided two major KPIs the prediction model needs to improve:

* Sby\_utilization: Percentage of standbys being activated is higher than in the current approach of keeping 90 drivers on hold
* Sby\_exceeded: Situations with not enough standbys should occur less often than in the current approach.

The Table 1 shows the base line KPIs for the whole dataset and for the last 365 days. The prediction model will be evaluated for the last 365 days thus the relevant KPIs are Sby\_utilization is 28,61% and Sby\_exceeded is 22,47%

Table 1: Base line KPIs

|  |  |  |
| --- | --- | --- |
|  | All data points | Last 365 days |
| Sby\_utilization | 20.43 % | 28.61 % |
| Sby\_exceeded | 14.84 % | 22.47 % |

# Model Training and Evaluation

## Model requirements and library selection

As in chapter 6 shown the given data is time series of a daily dataset with a weekly and annual seasonality. Besides the seasonality a trend is given. The prediction model should be capable to process weekly and annual seasonality as well as trends. 3 possible time series predictions models were identified being capable of performing the given task. The first is SARIMAX models. The SARIMAX model is a extension of the well known ARIMA (autoregressive integrated moving average) model which is able to directly handle seasonal data. As in chapter 6 described exists a weekly and an annual seasonality. SARIMAX can handle to seasonality with exogenous variables. Through one hot encoding each weekday gets a own column and will be fed to the model as the exogenous variable. The annual seasonality is described in the period parameter m (Brownlee 2019). Unfortunately the used pmdarima.auto\_arima tool for finding the right parameter (p, d, q, P, D, Q) is unable to process such long seasonal periods sufficiently (Hyndman 2010). A work around is to transform the annual seasonality information with Fourier and feed it as a exogenous variable to the auto\_arima model with a weekly seasonality (m=7) (see Stackoverflow 2021). This approach is successful.

Another well-developed open source library is Facebooks Prophet. Prophet is able to process large datasets of data of hourly, daily or weekly observations over a period from at least one year. It can consider special days – e.g. holidays – in its predictions and trend changes. Prophet is easy to use and delivers accurate results in high quality. An additive regression model is used with four main components: yearly seasonality is modeled by a Fourier series, weekly seasonality b dummy variables (one hot encoding), user provided list of holidays and a module to detect trend changes (see Taylor 2017A and Taylor 2017B).

A third module is the exponential smoothing state space model TBATS.

In the following model training Facebooks Prophet was chosen since the predictions models were the most accurate once for the given needs with the easiest customization.

## Model Training and evaluation

The dataset were split in a train and a test dataset. Since the annual seasonality is important the smallest test dataset is one year (365 days). The dataset contains in total 1152 rows. Thus the test data contains 31,7% of all values which is slightly higher than usual but still acceptable.

The given dataset needs to be transformed in a Prophet compatible format as a Pandas DataFrame with two columns “ds” for dates and “y” for the value on which Prophet will fit the model. As in chapter 6.2 explained exist a causation between the calls and the resulting number of drivers\_atwork. Therefore the variable calls is the “y” column.

There are several important parameter in the Prophet model. We have a linear growth model (default growth=’linear’). The daily, weekly and annual seasonality is analyzed automatically without further settings. The interval\_width provides an estimated guess based on historical data on how much & of the further datapoints will be between the lower and the upper predicted boundaries (“yhat\_upper” & “yhat\_lower”). Between the lower and upper bound is the prediction yhat of the calls (see (Prophet 2021 and Merwe 2018). For our given business problem following approach was chosen: The number of drivers on duty is estimated by the prediction variable “yhat” (predicted amount of calls) divided by the linear coefficient (4,82 calls per driver) as explained in chapter 6.2. The number of standby drivers (n\_sby) is predicted by the delta of calls “yhat\_upper” and “yhat” divided by the linear coefficient (4,82 calls per driver) as explained in chapter 6.2. The model was tested for different levels of interval\_width from 0.7 till 0.99. At interval\_width of 0.975 the KPIs indicator reaches for Sby\_exceeded 6,8% and for Sby\_utilization 44,4%. The KPI Sby\_utilization decreased with increasing interval\_width since the likelihood for exceeding the number of drivers in standby were minimized. Thus the percentage of sby\_exceeded increased as depicted in .

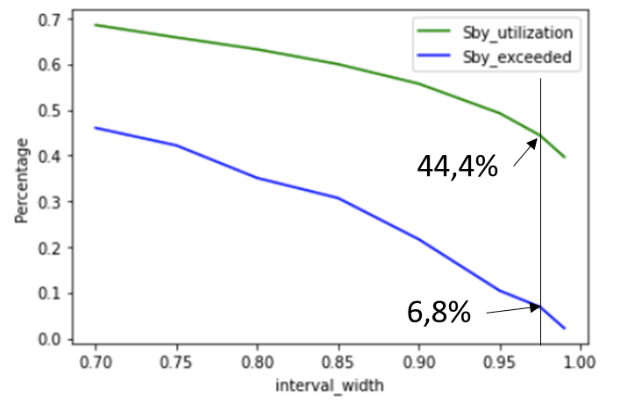


Figure 11: KPIs independence of interval width

The Figure 12 shows the model predictions over complete period at a interval\_width of 97.5% (number of drivers at work [black], predicted number of drivers on duty [blue], number of predicted drivers on duty + standby [red])

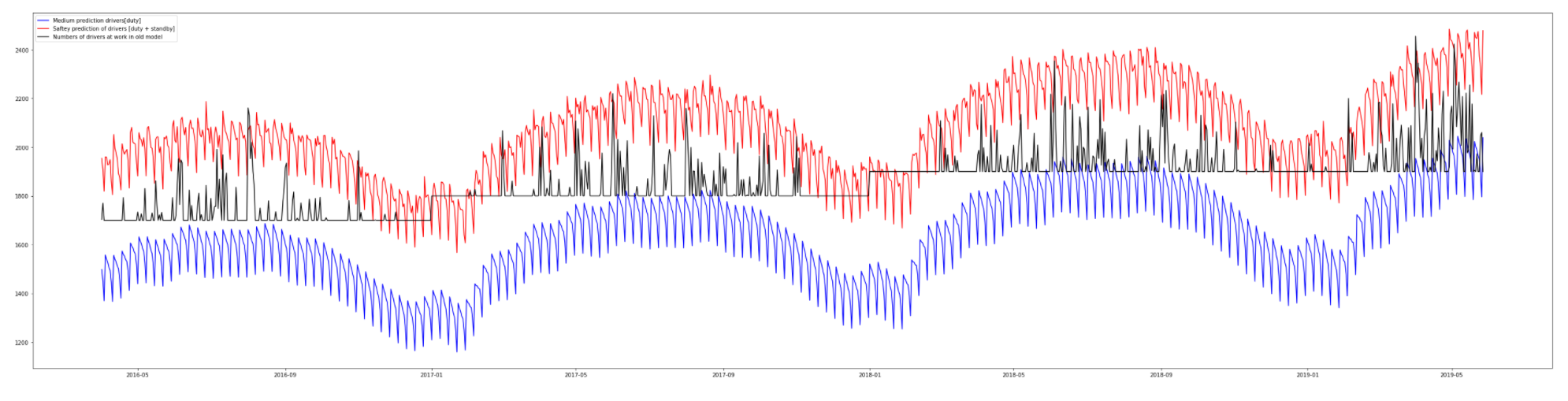


Figure 12: Model result over complete period at a interval\_width of 97.5% (number of drivers at work [black], predicted number of drivers on duty [blue], number of predicted drivers on duty + standby [red])

The root mean square error (RMSE) is 1012 and the Figure 13 shows the result of the cross validation. The mean absolute percentage error is between 8 and 12%.

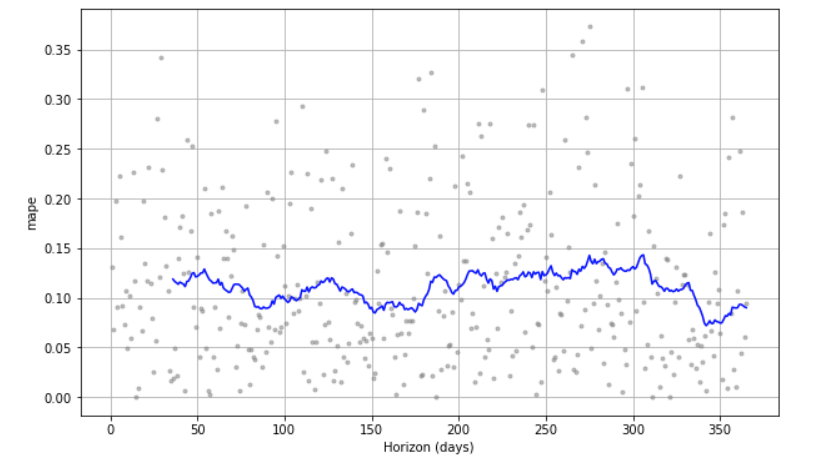


Figure 13: Prophets Diagnostic Cross Validation (Mean absolute percentage error)

# Deployment

A Jupyter notebook is created for the business to predict the needed drivers on duty (pred\_n\_duty) and the needed standby drivers (pred\_n\_sby) for the next 31 days. The model is trained by the whole datasets gathered till the date of use to improve the performance. The parameter are set as in chapter 7.2 described. An csv file is generated with the date of prediction and the pred\_n\_duty as well as pred\_n\_sby to support the fast adoption. An on the job training for at least two HR members is planned to facilitate an easy and safe start of the predictions. The HR member will have a key account partner where questions and possible issues can be answered/ resolved within 2 hours.

# Discussion and Conclusion

The aim of the project was to increase Sby\_utilization (Percentage of standbys being activated is higher than in the current approach of keeping 90 drivers on hold) and to decrease the amount of Sby\_exceeded (Situations with not enough standbys should occur less often than in the current approach). The described model increased the utilization of standby drivers from 28.61 % to 44.4%. The number of situations where the number of standbys were exceeded decreases from 22.47 % to 6.8%. Both KPIs were improved significantly.

Berliner Red-Cross provides services critical for social welfare. A key requirement for the success and the trust of the customers is a high reliability of the transport services. If not enough drivers are on duty or on standby the reliability will be impacted negatively. That’s why the given features benefiting the decrease of Sby\_exceeded over the utilization of standbys (in case of constant Sby\_exceeded a increase of Sby\_utilization to 55,8% is realistic). The parameter interval\_width is a key feature to influence the trade off between those two KPIs. The higher the interval\_width the unlikelier the amount of drivers on duty + standbys will be exceeded based on the historical dataset. With an increased interval\_width the amount of driver shifts increase as well (especially of the standby drivers) which will have a slightly negative cost effect. Since the business of Berliner Red-Cross is a critical infrastructure the reliability more important than saving money.

The mean average percentage of error for the predictions of the numbers of drivers on duty n\_duty calculated in the cross validation is between 8 and 12%. The planning based only on the number of drivers would be less accurate. Besides the number of drivers on duty the number of drivers on standby n\_sby is predicted. In the test time window of one year on 22 days the number of standbys and on duty drivers were exceeded. In the base model the number of on duty and standby drivers were exceeded at 88 days. This shows that the model is highly reliable and fulfills the business requirements to improve the utilization of standbys as well as reducing the percentage of days where the amount of drivers needed exceeds the amount of on duty drivers plus the amount of standby drivers.

# Library

* Microsoft-A (2021), Was ist der Team Data Science-Prozess (TDSP)?, <https://docs.microsoft.com/de-de/azure/architecture/data-science-process/overview>, last access: 18.11.2021 at 21:20
* Microsoft-B (2021), Lebenszyklus des Team Data Science-Prozesses, <https://docs.microsoft.com/de-de/azure/architecture/data-science-process/lifecycle>, last access: 18.11.2021 at 21:20
* Thakurta, D., McGehee, H. (2017), Team Data Science Process: Roles and tasks, <https://github.com/Azure/Microsoft-TDSP/blob/master/Docs/roles-tasks.md>, last access: 18.11.2021 at 21:21
* Deutesches Rotes Kreuz – Berlin (2021), <https://www.drk-berlin.de/>, last access 19.11.2021 at 17:58
* Brownlee, J. (2019), “ A gentle introduction to SARIMA for Time Series Forecasting in Python”, <https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/>, last access 27.11.2021 at 17:48
* Hndman, R. (2010), Forecasting with ling seasonal periods, <https://robjhyndman.com/hyndsight/longseasonality/>, last access at 27.11.2021 at 17:50
* Stackoverflow 2021, “Forecast time series with multiple seasonality by using aut\_arima(SARIMAX) and Fourier terms, <https://stackoverflow.com/questions/68923679/forecasting-time-series-with-multiple-seasonaliy-by-using-auto-arimasarimax-an>, last access 27.11.2021 at 18:21
* Taylor, S., Letham, B. (2017A), Forecasting at Scale, <https://doi.org/10.7287/peerj.preprints.3190v>
* Taylor, S., Letham, B. (2017B), Prohept: forecasting at scale – A Blog, <https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/>, last access at 27.11.2021 at 18:30
* Prophet (2021), “Uncertaintiy Intervals”, <https://facebook.github.io/prophet/docs/uncertainty_intervals.html>, last access 27.11.2021 at 20:41
* Merwe, R. (2018), Implementing Facebook Prophet efficiently, <https://towardsdatascience.com/implementing-facebook-prophet-efficiently-c241305405a3>, last access 27.11.2021 at 20:42