

**Course: Model Engineering DLMDSME01**

**CASE STUDY:**

**AUTOMATION OF STANDBY DUTY PLANNING FOR RESCUE DRIVERS VIA A FORECASTING MODEL**

**Author: Husam Romman**

**Matriculation: 32003216**

**Tutor: Dr. Sahar Qaadan**

**Delivery date: 25.4.2023**

# − Table of contents

# Introduction

One of the motivations why I chose this case study, is because I lived in Berlin about six years during my study long time ago.

This case study aims to present an automated way to forecast the standby duty plan for rescue drivers of the German Red Cross in the capitol Berlin.

Using the provided dataset, a machine learning model will be developed to estimate the daily need for standby rescue drivers in Berlin. Instead of keeping a fixed number of activated drivers in standby every day (the current situation), Our goal will be to have a higher percentage of standbys being activated and a lower number of situations with not enough standbys than in the current approach of keeping 90 drivers on hold.

I will rely on “The CRoss Industry Standard Process for Data Mining” (CRISP-DM) methodology in this use case, in which the Data Science Lifecycle go throw six major stations, namely:

* Business Understanding
* Data Understanding
* Data Preparation
* Modeling
* Evaluation
* Deployment

I used Anaconda as development environment with Jupyter Notebook for coding. The source code as many visualization contents will be pushed to GitHub.

Link:

**Case Study: Automation of Standby Duty Planning**

**for Rescue Drivers via a Forecasting Model**

# Background Information & Domain Knowledge

## **Demographics of Berlin:**

The city-state of Berlin had about 3,769 million registered residents in 2019 and covered an area of 891.82 square kilometers. Berlin is both the most populated city in the EU and the largest city in Germany (BerlinStatisticalFigures, 2019) .

A “Statista” graph shows that the residential population of Berlin was in 2016 was about 3.574 million (statista, n.d.). Which means no significant increase of population.

|  |
| --- |
|  |
| AREAS OF APPLICATION OF THE BERLIN FIRE DEPARTMENT  source:(BerlinerFeuerwehr, 2021) |

## **The German Red Cross:**

# The German Red Cross e. V. defines itself as: “the National Society of the Red Cross on the territory of the Federal Republic of Germany and a voluntary aid organization for the German authorities in the humanitarian field. It respects the principles of the International Red Cross and Red Crescent Movement” (DeutschesRotesKreuz, n.d.).

# Emergency Medical Service (EMS) is a service of public pre-hospital emergency healthcare, including ambulance service, provided by individual German cities and counties. Responding to all calls involving an urgent threat to a person's life or health is considered emergency services. This is the main aspect of the service, which is known in German as “Notfallrettung “or “Rettungsdienst”. This service deals with conditions involving sudden onset of disease and damage, including myocardial infarction and serious injury accidents, to name only two (Wikipedia-II, 2022).

## **Paramedics in Berlin:**

According to the last annual report from Berlin Fire Brigade (Berliner-Feuerwehr), there is now 1000 registered paramedics in Berlin (BerlinerFeuerwehr, 2021). The report speaks about a mission every 64 second in 2021, Average daily within 2,900 emergency calls processed 24 hours a day, “Those emergency calls led to more in 2021 than 490,000 missions”(BerlinerFeuerwehr, 2021).

# Task

## **Business understanding**

Business claims, that having a daily fixed number of standbys (n\_sby = 90) is not efficient because there are days with too many standbys followed by days with not enough standbys. The business aims at a more dynamical standby allocation, which takes seasonal patterns into account.

**Project Aim:**

Help the HR department with planning to estimate the amount of daily standby rescue drivers via a prediction model more efficiently. Here, efficient means that the percentage of standbys being activated is higher than in the current approach of keeping 90 drivers on hold. It also means that situations with not enough standbys should occur less often than in the current approach. Note that the plan must be finished on the 15th of the current month for the upcoming month.

**business objectives of the project:**

* To improve the current standby-duty plan.
* To provide a predictive model which can be used to estimate the daily need for standby rescue drivers.
* The goal is to have a higher percentage of standbys being activated and a lower number of situations with not enough standbys than in the current approach of keeping 90 drivers on hold.
* the model should minimize dates with not enough standby drivers at hand.

## **Data Understanding:**

The data source is a csv file (sickness\_table.csv) provided. This dataset contains about 3 years of daily information (from 2016 - 2019) on sickness counts, emergency calls, available standby resources and how many additional resources are activated.

**Explore the data:**

|  |
| --- |
|  |
| First look to the Dataset |

**Column Description:**

* **date**: entry date
* **n\_sick**: number of drivers called sick on duty
* **calls**: number of emergency calls
* **n\_duty**: number of drivers on duty available
* **n\_sby**: number of standby resources available
* **sby\_need**: number of standbys, which are activated on a given day
* **dafted**: number of additional drivers needed due to not enough standbys
* Data shape
  + Running (data.shape) tells us that we have 8 columns and 1152 entries.
* Column “Unnamed: 0” need to be dropped.
* Column date need to be converted to datetime type instead of object type.
* Checking for missing data gives no results.
* **EDA:**

As we can see, the dataset contains only numerical features, and the output variable (sby\_need) is given. Therefor we should use a supervised learning approach in this case.

I will use visual inspection to understand relationships.

* Create histograms of each variable, note their distribution, spread, or any outlier.

|  |
| --- |
|  |
| pairplot |

* Correlation

|  |
| --- |
|  |
| heatmap |

|  |
| --- |
|  |
| Correlation with standby needed |

**Inspection:**

**Feature drop:**

* “Standby needed” variable has high correlation with “calls” variable and “dafted” variable. Of course, the more calls received, the more rescue drivers are needed. Calls have nearly a Gaussian distribution.  
  Taking a look to the plot bellow

|  |
| --- |
|  |
| Figure 1 |

We notice that the activation of standby drivers starts approximately when calls number is greater than the mean. We can see also that calls increased linearly over three years, while the standby need was near constant under 35 drivers tell 2018, then started to increase as the calls number increased above its mean (around 8000 calls).

|  |
| --- |
|  |
|  |
|  |

* The “dafted” variable means the number of additional drivers needed due to not enough standbys, so  
   ,   
  we should keep this feature for now.
* Considering the increase of population in Berlin between 2016 and 2019 (about 100000) and number of duty drivers, which increased from 1700 to 1900 drivers, we should see – theoretically – a decrease of actual standby drivers needed, but this is not the case. I found a sudden increase of duty drivers on 1st January 2017, and 1st January 2018 by 1000 each. According to the “DRK – Jaresbuch” from 2019, the number of DRK active members in berlin was 1927 (DRK, 2019). This information means, the numbers are real. From the other side, no significant change in actual number of standby drivers for three years. Checking the correlation heatmap we see (0.091), therefor we can drop this feature (n\_duty).
* The number of sick drivers has a low correlation with needed standby (0.022), but it makes sense to keep this feature.
* The standby number feature can be dropped, sense it’s a fixed number (90 drivers).

**Feature extraction:**

* Sense it’s a sort of timeseries data, we need to extract some features like (Year, Month, Day, Day of the Week).
* We subtract the number of sick resources from the number of drivers available on duty and add the number of standby resources need on a given date we get the actual number of resources on duty on that day.

|  |
| --- |
|  |
|  |

Behavior during years:

## It is worth noting that in 2018, as the number of drivers increased, the number of patients decreased. But on the other hand, the number of reserve drivers maintained a semi-constant linear increase.

|  |  |
| --- | --- |
|  |  |
|  | |

**Baseline model:**

The baseline model could simply use a fixed number of standby drivers for every day, without considering any seasonal patterns or other factors that might affect the demand for standbys. For example, the baseline model could use a fixed number of 90 standby drivers every day, as suggested in the original requirements.

While this approach is simple and easy to implement, it is unlikely to be optimal because it does not consider any information about the actual demand for standby drivers on each day. This means that there may be many days where too many standby drivers are activated, leading to unnecessary costs, or not enough standby drivers are activated, leading to missed opportunities and reduced safety.

A better choice is to take the mean as a baseline

### Data Evaluation

As seen the data is structured data type, provided as a csv table

* Check head and tail:

data.head()

* Running data.columns gives us the columns name

Index(['Unnamed: 0', 'date', 'n\_sick', 'calls', 'n\_duty', 'n\_sby', 'sby\_need',

'dafted'],

dtype='object')

* Running data.shape tells us that we have 8 columns and 1152 entries.

(1152, 8)

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1152 entries, 0 to 1151

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 1152 non-null int64

1 date 1152 non-null object

2 n\_sick 1152 non-null int64

3 calls 1152 non-null float64

4 n\_duty 1152 non-null int64

5 n\_sby 1152 non-null int64

6 sby\_need 1152 non-null float64

7 dafted 1152 non-null float64

dtypes: float64(3), int64(4), object(1)

memory usage: 72.1+ KB

The first check is then to determine whether all fields are filled and whether the data conform to what is defined in the structure or schema.

We have one datatype as “object”, which is the date entry.

## Column Description:

• date: entry date

• n\_sick: number of drivers called sick on duty

• calls: number of emergency calls

• n\_duty: number of drivers on duty available

• n\_sby: number of standby resources available

• sby\_need: number of standbys, which are activated on a given day

• dafted: number of additional drivers needed due to not enough standbys

Checking for missing Data:

Drop column

data= data.drop("Unnamed: 0", axis = 1)

data.head()

date n\_sick calls n\_duty n\_sby sby\_need dafted

0 2016-04-01 73 8154.0 1700 90 4.0 0.0

1 2016-04-02 64 8526.0 1700 90 70.0 0.0

2 2016-04-03 68 8088.0 1700 90 0.0 0.0

3 2016-04-04 71 7044.0 1700 90 0.0 0.0

4 2016-04-05 63 7236.0 1700 90 0.0 0.0

# Understanding the Data

sickness\_table.csv contains daily information on sickness counts, emergency calls, available standby resources and how many additional resources are activated.

We subtract the number of sick resources from the number of drivers available on duty and add the number of standby resources need on a given date we get the actual number of resources on duty on that day.

df['actual\_duty'] = df['n\_duty'] - df['n\_sick'] + df['sby\_need']

Subtracting the needed standby resources from the standby number and adding that to the actual number of resources on duty on that day will give us the the whole number of needed resources.

df['actual\_duty\_and\_sby'] =df['actual\_duty'] + df['n\_sby'] - df['sby\_need']

Discussion:

Checking the percentage of activated drivers above n\_Sby=90 was 14.8 %

While the none zero percentage is 26.3%, that means only in about 26.3 % of the time standby drivers were activated. The rest of the time there was no need for standby drivers.

These facts can help us to

# git repository for the project:

**Deployment**[**¶**](http://localhost:8888/notebooks/Documents/IUBH/CASE%20STUDY-%20MODEL%20ENGINEERING/DLMDSME01-main/01_ds_methodology/01_ds_methodology.ipynb#Deployment)

**Customer acceptance**

**Setting up the git structure**

In accordance with the lifecycle stages of a data science project the git-structure will be:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

├── etl

│ ├── README.md

│ ├── notebooks

│ ├── docs

│ ├── src

│ │ ├── main.py

│ | ├── funcs

├── dev

│ ├── README.md

│ ├── notebooks

│ ├── docs

│ ├── src

│ │ ├── main.py

│ | ├── funcs

├── val

│ ├── README.md

│ ├── docs

│ ├── src

│ │ ├── main.py

│ | ├── funcs

├── depl

│ ├── README.md

│ ├── runscript.bat

│ ├── docs

│ ├── src

│ │ ├── main.py

│ | ├── funcs

├── README.md

├── package.bat

with

* **etl (extract-transform-load)**: here automation code related to data gathering, data warehouse should be created
* **dev (development)**: here the automated machine-learning model should be created
* **val (validation)**: here the final model should be (in-depth) validated
* **depl (deployment)**: here the final version of the machine-learning model adapted for the IT-infrastructure should be created

where

* **README.md** : describes the problem and the aim of each section
* **src** : is the automated source-code (with a main.py script)
* **docs** : are documents, like plots, presentations...
* **.bat (.sh) scripts** : are automation scripts
* **package.bat (.sh)** : creation of the python virtual environment and automated download of packages

# Conclusion

In this use case developed a machine learning model to forecast future standby duty rescue drivers for the Berliner Red Cross. Instead of keeping a fixed number of activated drivers in standby every day (the current situation). The goal is to have a higher percentage of standbys being activated and a lower number of situations with not enough standbys than in the current approach of keeping 90 drivers on hold.

CRISP-DM methodology is used in this use case, in which the Data Science Lifecycle go throw six major stations.

After many tries with time series methods at the beginning I used machine learning framework to develop the model.

The prediction results are very good with r-score = .

I used also Prophet and got also good results.

An accurate predictive model should consider seasonal patterns, weather conditions, holidays, and other relevant factors. The model would then dynamically allocate standby drivers based on the predicted demand, with the goal of maximizing the percentage of activated standby drivers while minimizing the number of days with not enough standbys.

Due to maximum page restrictions I pushed the source code, many visualizations to GitHub and provided the link of it.

I think this task pushed me to practice model engineering deeply from A to Z, which offered me a great opportunity to learn a lot.

# − Bibliography