

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: 13.5.2023

Table of Contents

[Introduction 3](#_Toc134911741)

[Background Information & Domain Knowledge 4](#_Toc134911742)

[**1.** **Demographics of Berlin:** 4](#_Toc134911743)

[**2.** **The German Red Cross:** 4](#_Toc134911744)

[**3.** **Paramedics in Berlin:** 4](#_Toc134911745)

[1. Business understanding 5](#_Toc134911746)

[2. Data Understanding: 5](#_Toc134911747)

[3. Data Preparation 6](#_Toc134911748)

[ **Feature Engineering** 10](#_Toc134911749)

[**1.** **Feature drop:** 10](#_Toc134911750)

[**2.** **Feature extraction:** 12](#_Toc134911751)

[3. Modelling 14](#_Toc134911752)

[4. Evaluation 15](#_Toc134911753)

[5. Deployment 18](#_Toc134911754)

[Conclusion 19](#_Toc134911755)

[Figure List 21](#_Toc134911756)

[Bibliography 21](#_Toc134911757)

# 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 due to limitation of pages.

Link: <https://github.com/Rom-Code/modelengineering/tree/a8d5144bcaf880f0783af49fce520f2e9be6674c/development>

# 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.

|  |
| --- |
|  |
| Figure 1: 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).

**Case Study: Automation of Standby Duty Planning  
for Rescue Drivers via a Forecasting Model**

# Business understanding

The objective of this project is to maximize the activation percentage of standby drivers while minimizing the risk of not having enough drivers on any given day. To achieve this, we will build a predictive model that will forecast the number of drivers needed for a given day based on historical data. The model should be able to accurately predict the number of drivers needed for each day, allowing the company to activate the optimal number of standby drivers and minimize the risk of not having enough drivers.

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.
* 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 set consists of historical records of the number of drivers needed for each day, along with various features that may affect the number of drivers needed. The data set contains 1152 records and 6 mean features.

The data source was provided as a csv file (sickness\_table.csv). 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:**

|  |
| --- |
|  |
| Figure 2: 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 Preparation

The first step in preparing the data for analysis is to perform exploratory data analysis (EDA) and visualization to better understand the patterns and trends in the data. This will involve investigating the relationships between variables and the additional features.

After the EDA is complete, feature engineering can be performed to create new variables that may be useful in predicting the number of drivers needed at a given time. For example, new variables may be created to capture the impact of holidays on driver availability.

* **Inspection**
  + Running (data.shape) tells us that we have 8 columns and 1152 entries and 8 columns.
  + Running (df.info) tells us more about columns and their data type:

|  |
| --- |
| * <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 |

* + 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.
* **Visualization ¶**

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.

Visual inspection let us understand relationships between variance features:

* + The line plot (Figure 3) shows seasonal pattern of calls as well as the standby needed and dafted. A small trend can be also recognized.
  + We can recognize two sudden increase number of drivers on duty available (n\_duty) at the beginning of 2017 and 2018, but there is no notable effect on other variables.

|  |
| --- |
|  |
| Figure 3: Line plot of features and variables |

* + Create histograms of each variable, to study their distribution, spread, or any outlier.[[1]](#footnote-1)

|  |
| --- |
|  |
| Figure 4: pair plot |

* **Checking correlation**

|  |
| --- |
|  |
| Figure 5-A: heatmap and correlation |

|  |
| --- |
|  |
| Figure 5-B: Correlation with standby needed |

## **Feature Engineering**

### **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 6: Relationship between Calls and standby needed |

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).

|  |
| --- |
|  |
|  |
| Figure 7: Calls and Standby needed graph over years |

* Behavior during years:

It is worth to mention, 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.

|  |  |
| --- | --- |
|  |  |
| Figure 8: Behavior during years | |

* The “dafted” variable means the number of additional drivers needed due to not enough standbys, so  
   ,   
  we could drop this feature.
* Considering the increase of population in Berlin between 2016 and 2019 (about 100000) and the 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 during the same time. Checking the correlation heatmap we see (0.091), sense no significant effect can be recognized, 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).
* Actual duty feature: 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.
* **Percentage:** although n\_sby feature was dropped because it’s a fixed number, a more dynamic feature can be added, namely the percentage[[2]](#footnote-2):

**Note**: This feature is highly correlated with sby\_need, therefor I will test prediction result with/without it.

|  |
| --- |
|  |
| Figure 9: Data Description Output |

* German holidays feature was also added.

|  |
| --- |
|  |
| Figure 10: Correlation Heatmap with extracted features |

The result of feature engineering was the following features:

['date', 'n\_sick', 'calls', 'actual\_duty', 'year', 'month', 'week', 'day', 'NumDayOfWeek', 'qaurter', 'percantage', , 'holidays\_de']

# Modelling

First, I split the data to a training (around 74%) and testing set (around 26%). Because the data is a sort of timeseries, I chose to define a fixed split date ('2018-08-01').

I used XGBoost (ReadTheDocs, n.d.) to build our predictive model. I first trained a baseline model using only two features (Calls and the number of sick drivers) as well as the default parameters and evaluated its performance using route mean squared error (RMSE). I then used all the features, optimized the hyperparameters of the model using cross-validation and grid search to improve its performance.

* **Baseline model:**

Many approaches can be used for the baseline model:

* We 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.
* Another choice is to take the mean of standby drivers as a baseline.
* While these two approaches are 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.

Using XGBoost I trained a baseline model using only two features (Calls and the number of sick drivers) as well as the default parameters and evaluated its performance using route mean squared error (RMSE).

**Results:**

|  |
| --- |
| [0] validation\_0-rmse:73.96831 validation\_1-rmse:113.30414  [100] validation\_0-rmse:35.43432 validation\_1-rmse:48.98381  [200] validation\_0-rmse:21.87021 validation\_1-rmse:35.12352  [300] validation\_0-rmse:16.62750 validation\_1-rmse:32.19929  [400] validation\_0-rmse:14.12642 validation\_1-rmse:31.77256  [500] validation\_0-rmse:12.75795 validation\_1-rmse:31.79312  [600] validation\_0-rmse:11.98703 validation\_1-rmse:31.86147  [700] validation\_0-rmse:11.38846 validation\_1-rmse:32.10565  [800] validation\_0-rmse:10.95588 validation\_1-rmse:32.40173  [900] validation\_0-rmse:10.46635 validation\_1-rmse:32.68593  [999] validation\_0-rmse:9.99244 validation\_1-rmse:32.91311 |

Using RMSE (Root Mean Squared Error) as an evaluation metric can be a good choice for several reasons, the most important for us is Interpretability; RMSE is on the same scale as the target variable, which makes it easier to interpret in the context of the problem. So, we can compare it with the mean of standby needed:

34.71875

We notice that baseline RMSE results are close to the mean of ‘sb\_need’, so we could use the mean as a baseline.

I then used all the features, optimized the hyperparameters of the model using cross-validation and grid search to improve its performance.

# Evaluation

We evaluated the performance of our model using various metrics such as RMSE, R score, MAE.

**Optimize Parameters using cross validation GridSearchCV():**

Parameters used (XGBoost, n.d.):

|  |  |
| --- | --- |
| 'min\_child\_weight' | #It defines the minimum sum of weights of all observations required in a child. |
| 'gamma' | #Gamma specifies the minimum loss reduction required to make a split. |
| 'subsample' | ,#It denotes the fraction of observations to be randomly samples for each tree. |
| 'colsample\_bytree' | #is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. |
| 'max\_depth': [2, 3, 4] | #The maximum depth of a tree |
| 'reg\_lambda': [1.5 , 2 , 3] | # L2 regularization term on weights. Increasing this value will make model more conservative. |

The evaluation results of XGBRegressor:

|  |
| --- |
| <bound method XGBModel.evals\_result of XGBRegressor  (base\_score=0.5, booster='gbtree', colsample\_bylevel=1,  colsample\_bynode=1, colsample\_bytree=1.0, gamma=0.3, gpu\_id=-1,  importance\_type='gain', interaction\_constraints='',  learning\_rate=0.300000012, max\_delta\_step=0, max\_depth=3,  min\_child\_weight=3, missing=nan, monotone\_constraints='()',  n\_estimators=100, n\_jobs=8, nthread=-1, num\_parallel\_tree=1,  random\_state=0, reg\_alpha=0, reg\_lambda=3, scale\_pos\_weight=1,  subsample=1.0, tree\_method='exact', validate\_parameters=1,  verbosity=None) |
|  |

METRICS RESULTS OF XGBOOST WITH "percantage" FEATURE:

r2\_score: 0.9936

MAE : 1.5709

RMSE : 8.1131

|  |
| --- |
|  |
| Figure 11: Forecast Plot vs Actuals |

|  |
| --- |
|  |
| Figure 12: Feature Importance |

We notice the dominant importance of the feature ‘percantage’, therefor I repeat model validation after dropping it to compare results.

Building a model after Dropping percentage feature due to high correlation:

|  |
| --- |
|  |
| Figure 13: Feature Importance - second |

METRICS RESULTS OF XGBOOST AFTER DROPPING "percantage" FEATURE:

r2\_score: 0.9624

MAE : 8.4828

RMSE : 19.7268

|  |
| --- |
|  |
| Figure 14: Best Parameters |

**Comparison: calculate the percentage of activated sby dirvers:**

|  |  |  |
| --- | --- | --- |
| Percentage in original data | Percentage in prediction | Percentage in prediction without 'percentage' feature |
| Mean = 38.576 | Mean = 58.05 | Mean = 59.755 |
| percentage above fixed number of sby = 90  mean\_orignial = 14.84  mean\_prediction = 5.729 | | |

# Deployment

The next step is to deploy the model into production and integrate it into the operations system. The model is used to make daily predictions of the number of drivers needed for the following month, allowing the Red cross to activate the optimal number of standby drivers.

To do that we need to develop a user-friendly graphical user interface (GUI) for the Standby Driver Optimization model to streamline the allocation of standby drivers. The GUI will provide predictions in the middle of the month for the next coming month, visualization of driver requirements, and optimization recommendations included to enhance decision-making.

**Proposed key features of the GUI:**

* Input Interface: Intuitive data input fields for historical driver availability, seasonal patterns, and relevant variables.
* Visualization: Graphical representation of predicted standby driver requirements over time.
* Threshold Setting: Adjustable activation threshold for calling standby drivers.
* Optimization Recommendations: Data-driven suggestions for the optimal number of standby drivers based on predicted demand.
* Scheduling Integration: Seamless integration with the HR department's scheduling system.
* Data Export: Ability to export prediction results and metrics for further analysis.
* User Support and Documentation: Clear instructions, tooltips, and comprehensive documentation for ease of use.

**Benefits:**

With Data-driven decisions by implementing a user-friendly GUI for the Standby Driver Optimization model, the business can optimize standby driver allocation, enhance operational efficiency, and ensure better service delivery with improved resource utilization.

**Git repository**

In accordance with the lifecycle stages of a data science project a recommended structure for the Git repository could be as follows (IU-DLMDSME01, 2021):

README.md: Overview of the project, including the problem statement, objectives, and key findings.

data/: Directory to store raw and processed data files.

notebooks/: Jupyter notebooks containing the code for EDA, data preprocessing, modeling, and evaluation.

src/: Source code files, including scripts for GUI development, model training, and prediction.

docs/: Documentation files, including user guides, data dictionaries, and model documentation.

results/: Directory to store model evaluation results, error analysis, and visualizations.

requirements.txt: List of Python libraries and dependencies required for running the project.

LICENSE: License file specifying the terms of use for the project.

.gitignore: File specifying files and directories to be ignored by Git.

By organizing the project in this manner, it becomes easier to navigate, collaborate, and maintain version control throughout the CRISP-DM process. The clear separation of code, data, documentation, and results ensures a well-structured and organized repository for effective project management.

# Conclusion

In this use case, I addressed the challenge of optimizing standby driver allocation for the HR department of the Berliner Red Cross. A predictive model is developed to estimate the daily standby driver requirements more efficiently. The model was trained on historical data, considering some seasonal patterns and other relevant variables. The given goal was 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 was used in this use case, in which the Data Science Lifecycle go throw six major stations.

After many tries with several approaches like time series, polynomial regression, random forest and Facebook Prophet algorithms at the beginning, I decided to use XGBoost to develop the model.

I discussed feature importance…

The metrics results are very good with:

r2\_score: 0.9624

MAE : 8.4828

RMSE : 19.7268

Our predictive model can successfully help the German Red Cross to maximize the activation percentage of standby drivers while minimizing the risk of not having enough drivers on any given day. The model accurately predicts the number of drivers needed for each day, allowing the Berliner Red Cross to activate the optimal number of standby drivers and minimize the risk of not having enough drivers.

Through the suggested graphical user interface (GUI), we can provide HR personnel with an intuitive tool to input data, visualize predictions, and make informed decisions. The GUI should allow threshold customization, optimization recommendations, and seamless integration with the scheduling system.

Additionally, we emphasized the importance of conducting a sophisticated error analysis to understand potential limitations and failure points of the approach. By analyzing feature importance, outliers, and temporal patterns, the business gains insights to refine and improve the model in the future. A more accurate predictive model should consider seasonal patterns, weather conditions, holidays, special big events in Berlin (like football games), and other relevant factors.

Overall, the Standby Driver Optimization model and GUI provide the Berliner Red Cross with a powerful tool to optimize standby driver allocation, enhance decision-making, and ensure efficient coverage. By leveraging data-driven insights, the Berliner Red Cross can improve its operational effectiveness, and resource allocation strategies.

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.

# Figure List

* Figure 1: AREAS OF APPLICATION OF THE BERLIN FIRE DEPARTMENT source:(BerlinerFeuerwehr, 2021)
* Figure 2: First look to the Dataset :
* Figure 3: Line plot of features and variables
* Figure 4: pair plot
* Figure 5-A: heatmap and correlation
* Figure 5-B: Correlation with standby needed
* Figure 6: Relationship between Calls and standby needed
* Figure 7: Calls and Standby needed graph over years
* Figure 8: Behavior during years
* Figure 9: Data Description Output
* Figure 10: Correlation Heatmap with extracted features
* Figure 11: Forecast Plot vs Actuals
* Figure 12: Feature Importance
* Figure 13: Feature Importance – second
* Figure 14: Best Parameters

# Bibliography

1. Azure, M. (n.d.). *What is the Team Data Science Process?* Retrieved from Microsoft Azure: https://learn.microsoft.com/en-us/azure/architecture/data-science-process/overview
2. BerlinerFeuerwehr. (2021). *JAHRESBERICHT 2021.* Retrieved from Berliner Feuerwehr: https://www.berliner-feuerwehr.de/fileadmin/bfw/dokumente/Publikationen/Jahresberichte/jahresbericht2021.pdf
3. BerlinStatisticalFigures. (2019). *Amt für Statistik Berlin Brandenburg*. Retrieved from Bevölkerungsprognose für Berlin und die Bezirke 2015 bis 2030: https://download.statistik-berlin-brandenburg.de/5bf355e1a1c6f379/6be4c820e27b/hz\_201904-02.pdf
4. DataScienceProcessAlliance. (2023, Jan 19). *What is CRISP DM?* Retrieved from Data Science Process Alliance: https://www.datascience-pm.com/crisp-dm-2/
5. DeutschesRotesKreuz. (n.d.). *DRK-Gesetz*. Retrieved from Deutsches Rotes Kreuz: https://www.drk.de/das-drk/auftrag-ziele-aufgaben-und-selbstverstaendnis-des-drk/drk-gesetz/
6. DRK. (2019). *DRK.de*. Retrieved from DRK Jahrbuch 2019 , page 57: https://www.drk.de/fileadmin/user\_upload/PDFs/Jahrbuecher/2019\_Jahrbuch/200430\_DRK\_Jahrbuch\_2019\_RGB\_RZ-Doppelseiten.pdf
7. IU-DLMDSME01. (2021). *01\_ds\_methodology.ipynb*. Retrieved from GitHub: https://github.com/iubh/DLMDSME01/blob/main/01\_ds\_methodology/01\_ds\_methodology.ipynb
8. ReadTheDocs. (n.d.). *XGBoost*. Retrieved from Python API Reference: https://xgboost.readthedocs.io/en/stable/python/python\_api.html
9. statista. (n.d.). *Development of the residential population of Berlin from 1995 to 2019* . Retrieved from Statista: https://www.statista.com/statistics/505892/berlin-population/#:~:text=This%20statistic%20shows%20the%20development,3.64%20million%20the%20previous%20year.
10. Wikipedia. (2022, Dec 13). *German Red Cross* . Retrieved from Wikipedia: https://en.wikipedia.org/wiki/German\_Red\_Cross
11. Wikipedia-II. (2022, Oct 22). *Emergency medical services in Germany* . Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Emergency\_medical\_services\_in\_Germany
12. Wikipedia-III. (2022, Aug 21). *Paramedics in Germany*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Paramedics\_in\_Germany
13. XGBoost, d. (n.d.). *XGBoost Parameters*. Retrieved from dmlc XGBoost: https://xgboost.readthedocs.io/en/stable/parameter.html

1. The rest of plots can be seen in the Jupyter notebook or separately on GitHub. Link: <https://github.com/Rom-Code/modelengineering/tree/main/development/images> [↑](#footnote-ref-1)
2. 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. [↑](#footnote-ref-2)