INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

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: 16.5.2023

Table of Contents

Introduction

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 for one month ahead. 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 **CR**oss Industry **S**tandard **P**rocess for **D**ata **M**ining” (**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 a development environment with Jupyter Notebook for coding. The source code as many visualization contents will be pushed to GitHub due to of page limitation.
Link: https://github.com/Rom-Code/modelengineering/tree/a8d5144bcaf880f0783af49fce520f2e9be6674c/development

Background Information & Domain Knowledge

1. 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.). This means, there was no significant increase of population between 2016 and 2019.

Figure 1: AREAS OF APPLICATION OF THE BERLIN FIRE DEPARTMENT
source: (BerlinerFeuerwehr, 2021)

2. The German Red Cross:

The German Red Cross e. V. defines itself ” (DeutschesRotesKreuz, n.d.) 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).

3. Paramedics in Berlin:

According to the last annual report from Berlin Fire Brigade (Berliner-Feuerwehr), there was 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

1. Business understanding

The objective of this project is to maximize the activation percentage of standby drivers of BRC, 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 each day for one month ahead, 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. 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 for one month.
- 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.

2. 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 main features.

The data source was provided as a csv file (sickness_table.csv) [1]. This dataset contains about 3 years of daily records (from 2016 - 2019) on sickness counts, emergency calls, available standby resources and how many additional resources are activated.

[1] https://github.com/Rom-Code/modelengineering/blob/cd39ea83d5e256369fb92f8e8be2876330a0105b/development/Data/sickness_table.csv

- **Explore the data:**

	Unnamed: 0	date	n_sick	calls	n_duty	n_sby	sby_need	dafted
0	0	2016-04-01	73	8154.0	1700	90	4.0	0.0
1	1	2016-04-02	64	8526.0	1700	90	70.0	0.0
2	2	2016-04-03	68	8088.0	1700	90	0.0	0.0
3	3	2016-04-04	71	7044.0	1700	90	0.0	0.0
4	4	2016-04-05	63	7236.0	1700	90	0.0	0.0

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.

- **Inspection**
  - Data description:

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	1152.0	575.500000	332.698061	0.0	287.75	575.5	863.25	1151.0
n_sick	1152.0	68.808160	14.293942	36.0	58.00	68.0	78.00	119.0
calls	1152.0	7919.531250	1290.063571	4074.0	6978.00	7932.0	8827.50	11850.0
n_dutv	1152.0	1820.572917	80.086953	1700.0	1800.00	1800.0	1900.00	1900.0
n_sbv	1152.0	90.000000	0.000000	90.0	90.00	90.0	90.00	90.0
sbv_need	1152.0	34.718750	79.694251	0.0	0.00	0.0	12.25	555.0
dafted	1152.0	16.335938	53.394089	0.0	0.00	0.0	0.00	465.0

  - 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
```

3. 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**
  - 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.
  - 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 (no missing data).

```
df.notna().count()
Unnamed: 0    1152
date          1152
n_sick        1152
calls         1152
n_duty        1152
n_sby         1152
sby_need      1152
dafted        1152
dtype: int64
```

- **Visualization ¶**

Visual inspection let us understand relationships between different features:

- (Figure 3): the line plot shows seasonal pattern of calls as well as the standby needed and dafted. A small trend can be recognized also.
- We can recognize two sudden increase in the number of drivers on duty available (n_duty) at the beginning of 2017 and 2018, but there is no notable effect on other features or the label (sby_need).

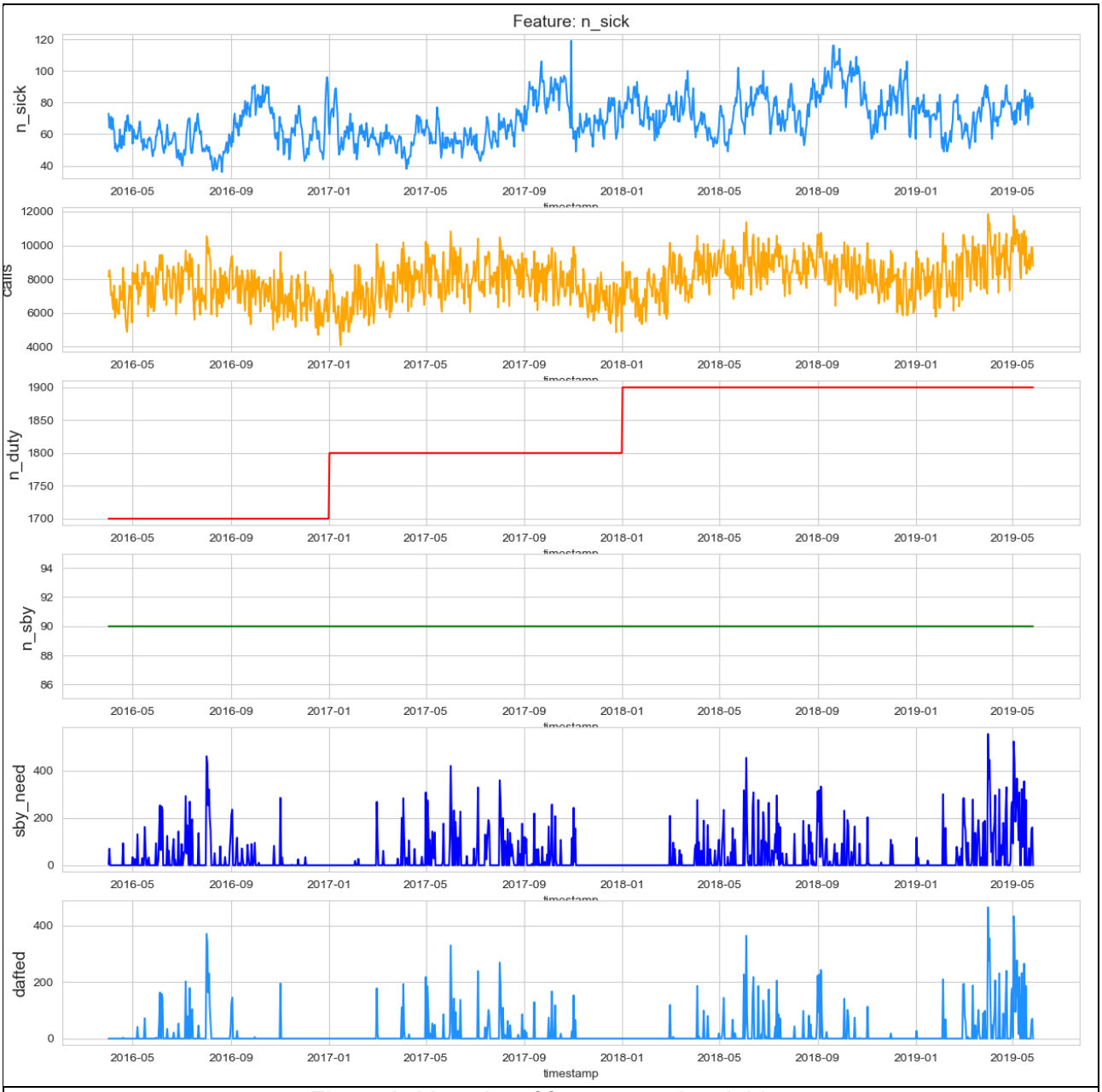

Figure 3: Line plot of features and variables

- A histogram of each variable was plotted, to study their distribution, spread, or any outlier (see Github).[2]

[2] 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

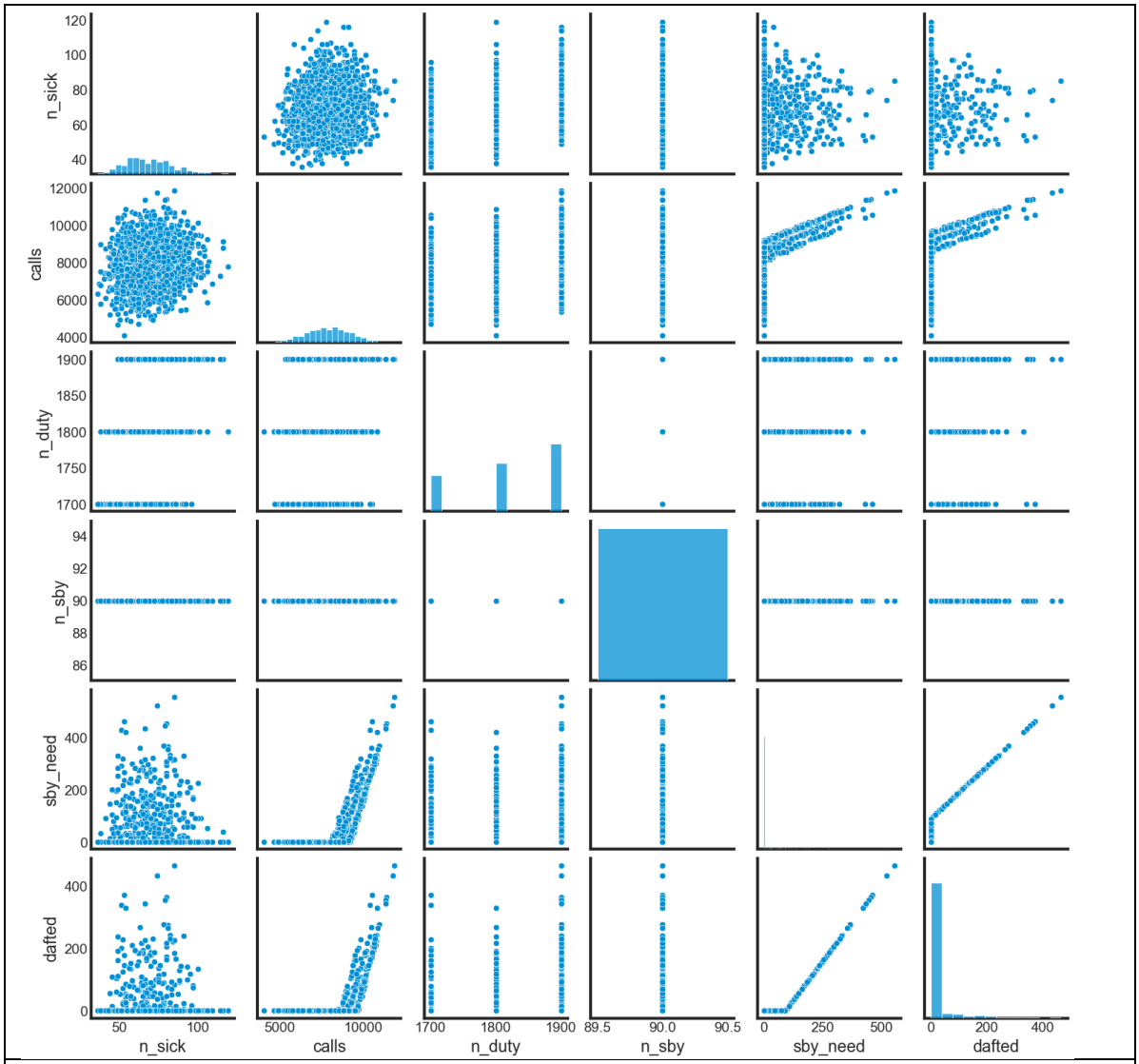

Figure 4: pair plot

- **Checking correlation**

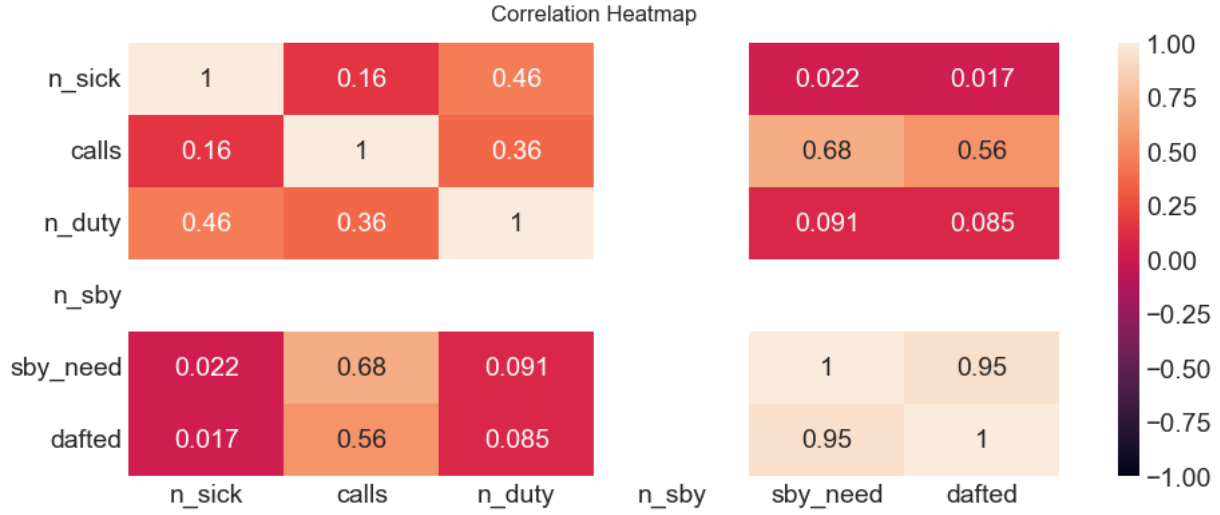

Figure 5-A: heatmap and correlation

	sby_need
sby_need	1.000000
dafted	0.945168
calls	0.677468
n_duty	0.090654
n_sick	0.022321
n_sby	NaN

Figure 5-B: Correlation with standby needed

According to the heatmap and correlation results we can do feature engineering.

- **Feature Engineering**

1. 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 (see Github)[3].

$$4074 \leq number\ of\ Calls\ a\ day \leq 11850$$

$$The\ mean\ is: 7919\ calls$$

$$The\ median\ is: 7932$$

Figure 6 shows the relationship between "calls" and "sby_need":

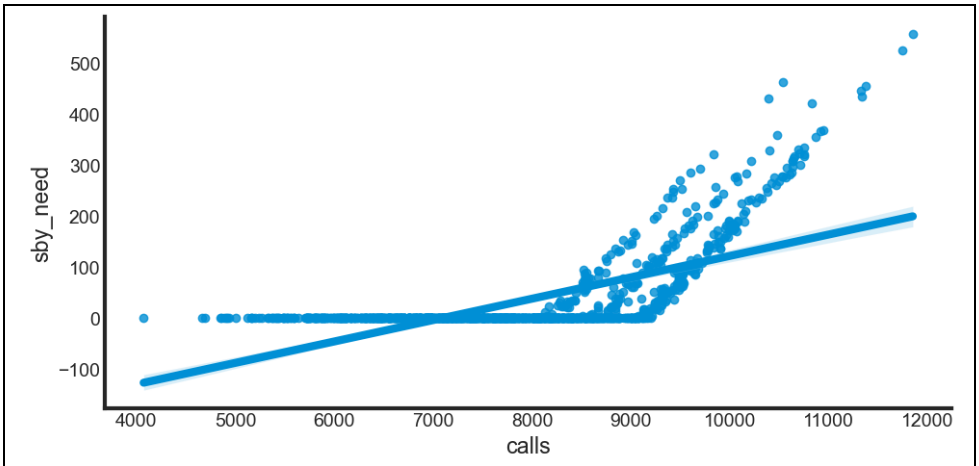

Figure 6: Relationship between Calls and standby needed

[3] 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

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), see figure 7.

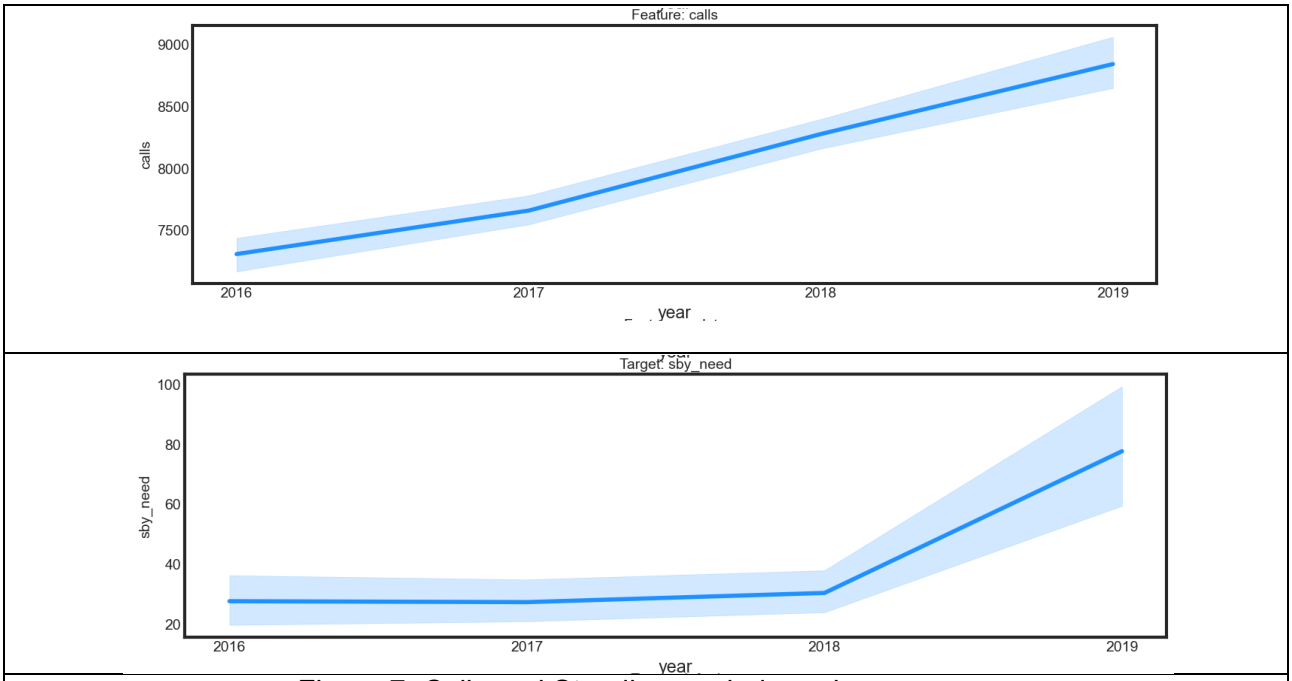

Figure 7: Calls and Standby needed graph over years

- Data behavior over four years:

It's worth to mention, that in 2018, as the number of drivers increased, the number of sick drivers decreased. But on the other hand, the number of reserve drivers maintained nearly its constant linear shape (Figure 8).

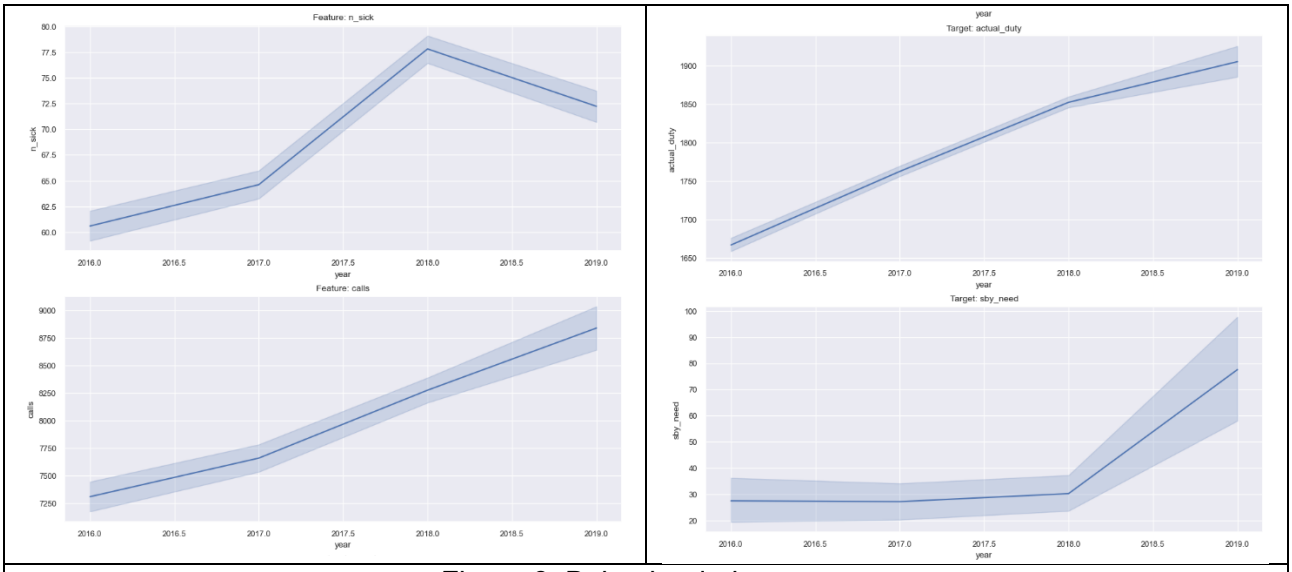

Figure 8: Behavior during years

- The "dafted" variable means the number of additional drivers needed due to not enough standbys, this feature is highly correlated with the label (sby_need), we could drop this feature.

$$dafted = \begin{cases} 0\ ,\ when\ needed\ drivers \leq n\_sby \\ sby_{need} - n\_sby\ ,\ when\ needed\ drivers > n\_sby \end{cases}$$

- Considering the increase of population in Berlin between 2016 and 2019 (about 100000 new residents) and the number of duty drivers, which increased from 1700 to 1900 drivers, we should expect 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 BRC active members in berlin was 1927 (DRK, 2019). This hint means, the given numbers in data set 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 the value of (0.091), sense no significant effect can be recognized, we can drop this feature ("n_duty").
- The number of sick drivers ("n_sick") has a low correlation with needed standby (0.022), but it makes sense to keep this feature, because someone should make up for the shortage of drivers.
- The standby number ("n_sby") feature can be dropped, sense it's a fixed number (90 drivers).

2. Feature extraction:

- Sense it's a sort of timeseries data, we need to extract some features like (Year, Month, Day, Day of the Week, Quarter).
- Actual duty feature: if we subtract the number of sick resources from the number of drivers available on duty and add the number of standby resources needed on a given date, we get the actual number of resources on duty on that day.

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

- **Percentage:** although n_sby feature was dropped because it's a fixed number, a more dynamic feature can be added, namely the percentage[4]:

$$percantage = (data.sby\_need\ /\ df.n\_sby) * 100$$

$$percantage.mean \approx 38.6$$

[4] Checking the percentage of activated drivers above n_Sby=90 gave us 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.

Note: This feature is highly correlated with sby_need (see Figure 9), therefor I will test the prediction result with/without it later.

```
[43]: data.describe().T
```

[43]:

	count	mean	std	min	25%	50%	75%	max
n_sick	1152.0	68.808160	14.293942	36.0	58.00	68.0	78.000000	119.000000
calls	1152.0	7919.531250	1290.063571	4074.0	6978.00	7932.0	8827.500000	11850.000000
actual_duty	1152.0	1786.483507	114.121491	1604.0	1721.75	1796.0	1834.000000	2370.000000
year	1152.0	2017.333333	0.977594	2016.0	2017.00	2017.0	2018.000000	2019.000000
month	1152.0	6.424479	3.394101	1.0	4.00	6.0	9.000000	12.000000
week	1152.0	26.023438	14.770423	1.0	14.00	25.0	39.000000	52.000000
day	1152.0	15.674479	8.778030	1.0	8.00	16.0	23.000000	31.000000
NumDayOfWeek	1152.0	3.002604	2.002386	0.0	1.00	3.0	5.000000	6.000000
qaurter	1152.0	2.484375	1.094858	1.0	2.00	2.0	3.000000	4.000000
holidays_de	1152.0	0.025174	0.156720	0.0	0.00	0.0	0.000000	1.000000
percantage	1152.0	38.576389	88.549168	0.0	0.00	0.0	13.611111	616.666667
sby_need	1152.0	34.718750	79.694251	0.0	0.00	0.0	12.250000	555.000000

Figure 9: Data Description Output

- German holidays feature was also added.
- Figure 10 shows the correlation heatmap with the extracted features:

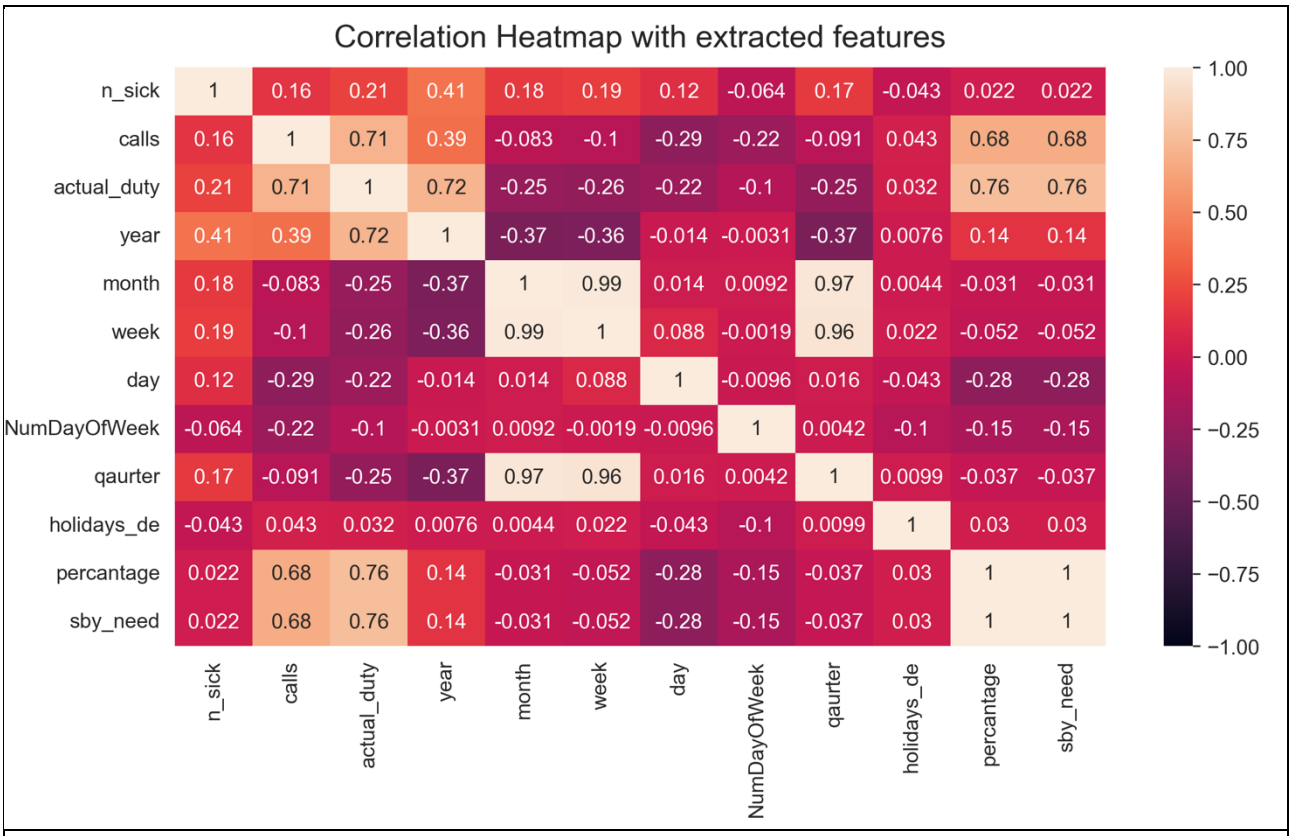

Figure 10: Correlation Heatmap with extracted features

- The resulting dataset of feature engineering will contain these columns:

['date', 'n_sick', 'calls', 'actual_duty', 'year', 'month', 'week', 'day', 'NumDayOfWeek', 'qaurter', 'percantage', , 'holidays_de']

3. 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 a predictive model. XGBoost is an advanced gradient boosting algorithm that generally provides better results than traditional machine learning algorithms. It is often used for regression, classification and ranking problems.

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.

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).

$$reg\_base = XGBRegressor(n\_estimators = 1000, early\_stopping\_rounds = 50, learning\_rate = 0.01)$$

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:

$$standby\ needed.mean(\quad) = 34.71875$$

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.

4. 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)
```

- **Best Parameters:**

```
[508]: grid.best_params_

[508]: {'colsample_bytree': 1.0,
        'gamma': 0.3,
        'max_depth': 3,
        'min_child_weight': 3,
        'reg_lambda': 3,
        'subsample': 1.0}
```

- **METRICS RESULTS OF XGBOOST WITH "percantage" FEATURE:**

r2_score: 0.9936

MAE : 1.5709

RMSE : 8.1131

The R2 score, also known as the coefficient of determination, measures the proportion of the variance in the target variable that is predictable from the input features. The closer the R2 score is to 1, the better the model's performance in terms of capturing the variance in the target variable based on the input features (Agrawal, 2022).

The lower the RMSE and MAE values, the better the model's performance in terms of prediction accuracy.

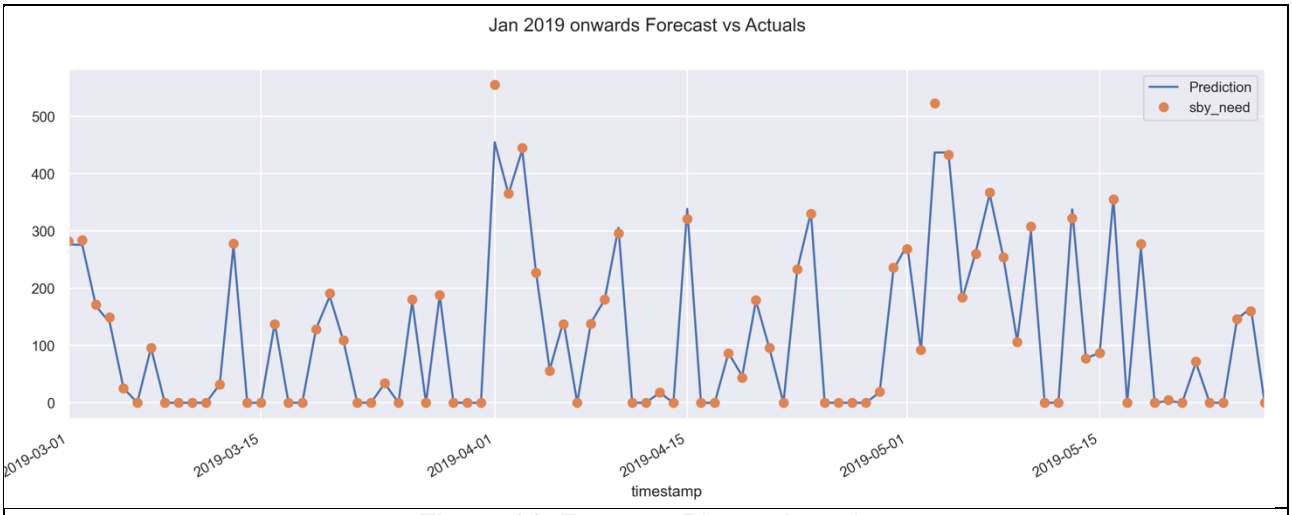

Figure 11: Forecast Plot vs Actuals

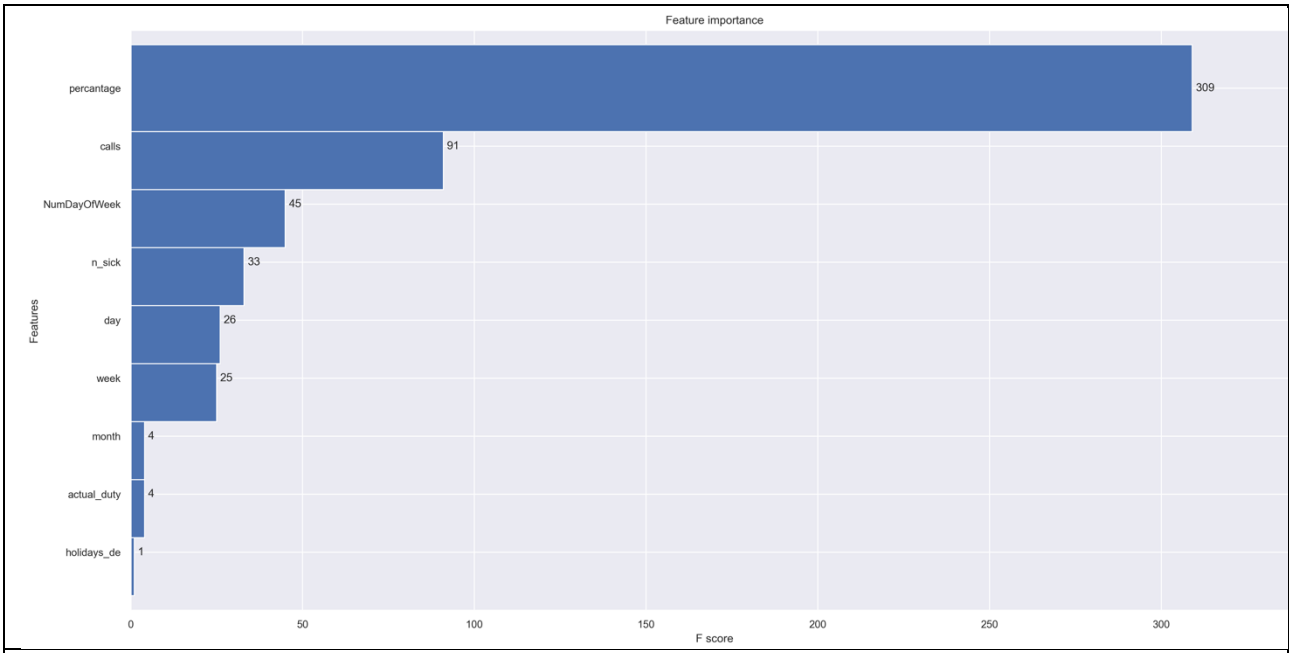

Figure 12: Feature Importance

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:

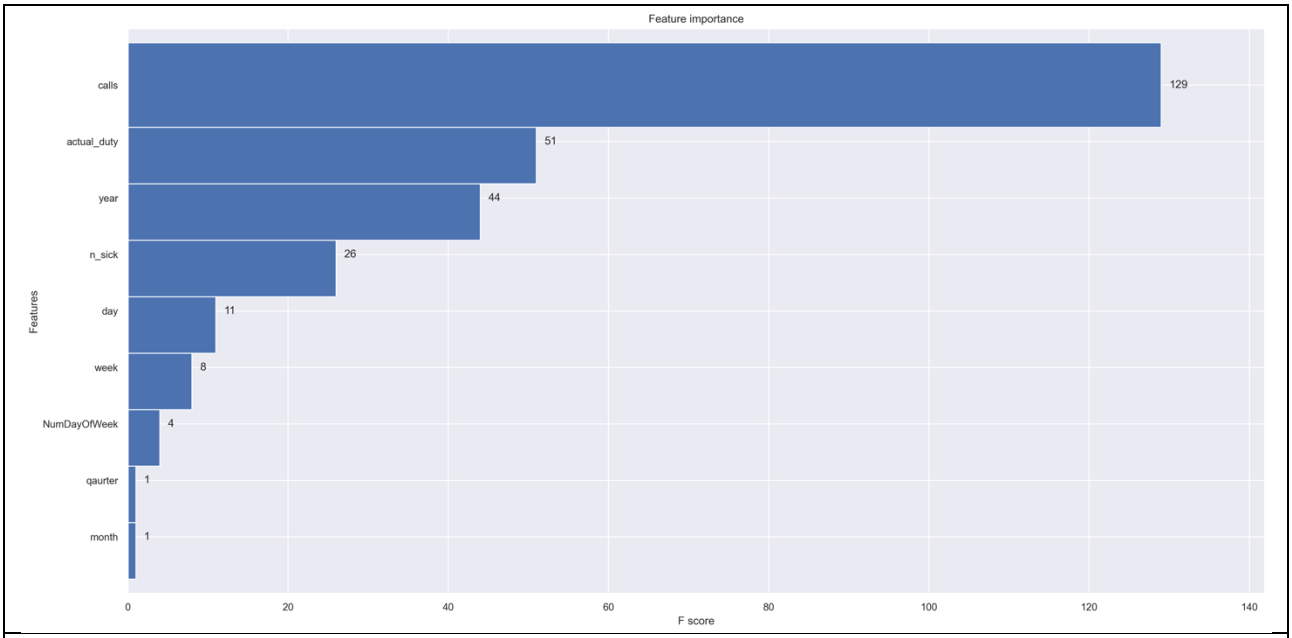

Figure 13: Feature Importance - second

- **METRICS RESULTS OF XGBOOST AFTER DROPPING "percantage" FEATURE:**

  r2_score: 0.9624

  MAE : 8.4828

  RMSE : 19.7268

```
[442]: grid_nop.best_params_

[442]: {'colsample_bytree': 0.8,
        'gamma': 0.1,
        'max_depth': 2,
        'min_child_weight': 3,
        'reg_lambda': 1.5,
        'subsample': 0.8}
```

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 **Our prediction model can provide lower percentage of standby drivers bellow 90.**		

5. 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 following (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 for one month 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 every day. This goal was reached.

CRISP-DM methodology was used in this use case, in which the Data Science Lifecycle go throw six major stages. 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 thoroughly and chose the best ones to build the model with. The evaluation metrics results are very good with:

r2_score: 0.9624 MAE: 8.4828 RMSE: 19.7268

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

Overall, the Standby Driver Optimization model and the suggested 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 (Link to my GitHub repository)[5].

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… thank you.

[5] https://github.com/Rom-Code/modelengineering/tree/a8d5144bcaf880f0783af49fce520f2e9be6674c/development

Figure List

Bibliography

i. Agrawal, R. (2022, July 21). *Know The Best Evaluation Metrics for Your Regression Model !* Retrieved from www.analyticsvidhya.com: https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

ii. 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

iii. BerlinerFeuerwehr. (2021). *JAHRESBERICHT 2021*. Retrieved from Berliner Feuerwehr: https://www.berliner-feuerwehr.de/fileadmin/bfw/dokumente/Publikationen/Jahresberichte/jahresbericht2021.pdf

iv. 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

v. DataScienceProcessAlliance. (2023, Jan 19). *What is CRISP DM?* Retrieved from Data Science Process Alliance: https://www.datascience-pm.com/crisp-dm-2/

vi. 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/

vii. 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

viii. IU-DLMDSME01. (2021). *01_ds_methodology.ipynb*. Retrieved from GitHub: https://github.com/iubh/DLMDSME01/blob/main/01_ds_methodology/01_ds_methodology.ipynb

ix. ReadTheDocs. (n.d.). *XGBoost*. Retrieved from Python API Reference: https://xgboost.readthedocs.io/en/stable/python/python_api.html

x. 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.

xi. Wikipedia. (2022, Dec 13). *German Red Cross* . Retrieved from Wikipedia: https://en.wikipedia.org/wiki/German_Red_Cross

xii. Wikipedia-II. (2022, Oct 22). *Emergency medical services in Germany* . Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Emergency_medical_services_in_Germany

xiii. Wikipedia-III. (2022, Aug 21). *Paramedics in Germany*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Paramedics_in_Germany

xiv. XGBoost, d. (n.d.). *XGBoost Parameters*. Retrieved from dmlc XGBoost: https://xgboost.readthedocs.io/en/stable/parameter.html