\section{Classification Models}

In accordance with the KDD process, different models for finding patterns in the pre-processed data are considered. The three different models presented in the following chapters are chosen based on previous publications regarding predictive quality classification in production engineering and recent trends in machine learning.

\subsection{Logistic Regression Model}

Logistic Regression is a method for the binary classification (2 classes) of an arbitrary input X\_i. It uses the logistic function **σ** (sigmoid function)to transform the continuous output of a Linear Regression Model to values between 0 and 1. This output is equivalent to the predicted probability P\_i of the input X\_i belonging to the class y\_i (falls platz hier die gleichung).   
The final discrete classification is based on the whether the probability P\_i is great or equal/smaller than 0.5.

For fitting the Logistic Regression Model, the regression coefficients of the linear model must be estimated. An ideal fit is achieved by maximizing the likelihood function of the model via iterative gradient-based methods. In the presented case the Limited-memory Broyden-Fletcher-Goldfarb-Shannon (LBFGS) optimizer is used. It is based on Newton’s Method, however computationally less expensive, which is critical, considering the limited computational resources at hand. \cite{

\subsection{Gradient boosting}

Gradient boosting is a recent machine learning technique used for regression and classification tasks based on ensemble learning. Multiple weak prediction models are iteratively generated based on the gradient of the model errors. The final strong prediction model is obtained by calculating the weighted average of all weak model. “XGBoost”, a gradient boosting method based on decision trees is chosen based on its performance in machine learning competitions and its computational efficiency. \cite{ [What is XGBoost? | Data Science | NVIDIA Glossary](https://www.nvidia.com/en-us/glossary/data-science/xgboost/), [XGBoost | Kaggle](https://www.kaggle.com/code/dansbecker/xgboost/notebook)}

The hyperparameters of the “XGBoost” gradient boosting model are:

* The number of iterations in the model generation cycle
* The learning rate/the weight of each model for the final prediction

Considering the limited computational resources at hand they are optimized using a randomized non-exhaustive grid search. For the final model the number of model generation cyles is set to XXXX and the learning rate to XXXX.

Furthermore, we performed a principal component analysis to improve in calculation speed and complexity.

\subsection{ Neural Network}

Another kind of model which can be used for the classification of data are Neural Networks (NNs). They consist of an input layer, multiple hidden layers and an output layer. The number of neurons in the hidden layers is arbitrary while in the input and output layer it corresponds with the dimension of the input/output. Depending on the task of the neural network (Regression or Classification), the so-called activation function in the output layers is chosen. The activation function determines the mathematical operations performed at each node. For the hidden layers the choice of the activation function is independent of the application of the NN.

For a NN, a large number of hyperparameters must be determined. The main ones are:\newline\newline

- Optimizer and Learning rate\newline\newline

- Activation function\newline

- Architecture of the NN (number of hidden layers and respective number of neurons)\newline

Based on engineering practices presented in the courses “AI in Production Engineering” and “Physics-informed machine learning”, as well as the paper “Real-time prediction of…”, the widely used ADAM optimizer is chosen together with a learning rate of 0.001.

Since a binary classification must be conducted, the “Sigmoid” activation function is used in the output layer.

For the activation functions of the hidden layers, the hyperbolic tangent (tanh) and the Rectified Linear Unit (ReLu) function are considered. Their performance with different NN architectures (1-7 layers, 5 – 100 neurons) is evaluated in multiple tests using a cross validation. The results show that especially with bigger NNs, a higher mean prediction accuracy can be achieved using the ReLu function. However, this comes at the expense of a higher variance and a gradual decrease in the share of false negatives/a gradual increase in the share of false positives. This indicates at least a partial overfit of the model on the dataset. Considering the planned application of the model, an emphasis is put on the robustness of the expected prediction and the prevention of a bias due to overfitting. Therefore, a model architecture with only one hidden layer and 75 neurons, using the tanh activation function is chosen.