

Universität Duisburg-Essen · Campus Duisburg

Fakultät für Ingenieurwissenschaften

Abteilung Maschinenbau

Lehrstuhl für Produktentstehungsprozesse und Datenmanagement

Masterarbeit

zur Erlangung des Grades eines

**Master of Science Automation and Safety (ISE)**

Thema: **Analyse und Einsatz von Methoden der künstlichen Intelligenz zur Identifikation von Anforderungen in Normendokumenten**

**Analysis and Deployment of artificial Intelligence Methods for Identification of Requirements in normative Documents**

vorgelegt von: **Mohamed Alazzazy**

geb. am 15.07.1995 in Egypt

Matrikel-Nr.: 309125

Erstgutachter: **Prof. Dr.-Ing. A. Nagarajah**

Zweitgutachter: **XXXX**

Betreuer: **M. Sc. Janosch Luttmer**

Duisburg, 05.08.2023

# Aufgabenstellung

# Kurzfassung

Normen sind eine wichtige Wissensquelle für die Produktentwicklung. In einer zunehmend digitalisierten Welt ist die Menge an normativen Dokumenten, die verschiedene Branchen und Sektoren regeln, exponentiell gewachsen. Diese Dokumente, die sich häufig durch ihre Komplexität und Länge auszeichnen, spielen eine entscheidende Rolle bei der Gestaltung von Richtlinien, Vorschriften und Normen. Die genaue Identifizierung von Anforderungen in solchen normativen Dokumenten ist nicht nur eine arbeitsintensive Aufgabe, sondern auch anfällig für menschliche Fehler. Diese Masterarbeit befasst sich mit dem Gebiet der künstlichen Intelligenz (KI), um diese Herausforderung zu bewältigen.

Das Hauptziel dieser Arbeit ist die Analyse und der Einsatz von KI-Methoden für die automatische Identifizierung von Anforderungen in normativen Dokumenten. Durch den Einsatz von Techniken der Verarbeitung natürlicher Sprache (NLP), Algorithmen des maschinellen Lernens und fortschrittlicher Datenanalyse zielt diese Studie darauf ab, ein robustes und effizientes System zu entwickeln, das Anforderungen aus einer Vielzahl von normativen Dokumenten analysieren und extrahieren kann.

**Schlüsselwörter**: Standarddokumente, künstliche Intelligenz, natürliche Sprachverarbeitung, maschinelles Lernen, Anforderungen, Analyse, Identifikation, Produktentwicklung, Masterarbeit.

# Abstract

Standards are an important source of knowledge in product development. In an increasingly digitized world, the volume of standard documents governing various industries and sectors has grown exponentially. These documents, often characterized by their complex and lengthy nature, play a critical role in shaping policies, regulations, and standards. The accurate identification of requirements within such normative documents is not only a labor intensive task but also prone to human error. This master's thesis delves into the realm of artificial intelligence (AI) to address this challenge.

The primary objective of this master thesis is to analyze and deploy AI methods for the automated identification of requirements in standard documents. By leveraging natural language processing (NLP) techniques, machine learning algorithms, and advanced data analysis, this study aims to improve a robust and efficient system that can parse and extract requirements from a diverse range of normative documents.

**Keywords:** standard documents, artificial intelligence, natural language processing, machine learning, identification, analysis, requirements, product master thesis.

# List of Contents

Aufgabenstellung I

Kurzfassung II

Abstract II

List of Contents III

List of Figures V

List of Tables V

List of Abbreviations VI

Symbolverzeichnis VII

1 Introduction 1

1.1 Problem Description 2

1.2 Goal of the Thesis 3

1.3 Structure of thesis 3

2 Theoretical foundation 4

2.1 Smart standard 4

2.1.1 Core elements in smart standards 4

2.1.2 Information extraction 6

2.1.3 Information extraction techniques 7

2.2 Text extraction 7

2.2.1 Text extraction pipeline: - 7

2.2.2 Text extraction techniques: - 8

2.3 Artificial Intelligence: - 9

2.3.1 Artificial Intelligence techniques: - 9

2.3.2 Text classification in artificial intelligence: - 9

3 State of the art 10

3.1 Parameters and Hyperparameters 10

3.1.1 Neural Network (NN) and Convolutional (CNN) 10

3.1.2 Support Vector Machines (SVM) 10

3.2 Hyperparameter tuning techniques 11

3.2.1 Manual trial-and-error Tuning 11

3.2.2 Grid Search 11

3.2.3 Random Search 11

3.2.4 Bayesian Optimization 11

3.2.5 Genetic Algorithms 11

3.2.6 Gradient-Based Optimization 11

3.2.7 Ensemble Methods 11

3.2.8 Automated Machine Learning (AutoML) 11

3.2.9 Cross-Validation 11

4 Methodology 12

5 Analysis 12

6 Concept development 12

6.1 Initial combination: - 12

6.2 Hyperparameters tuning: - 12

6.3 Advanced combination: - 12

7 Development and Implementation: - 13

8 Conclusion: - 14

9 Erster Abschnitt des Hauptteils 15

9.1 Hinweise zur Formatierung 15

9.2 Allgemeine Hinweise zur Benutzung von Word 17

9.2.1 Makros 17

9.2.2 Querverweise 17

10 Abbildungen und Tabellen 18

10.1 Verwendung von Abbildungen 18

10.2 Verwendung von Tabellen 20

11 Formeln 22

12 Zitieren mit Citavi 23

13 Druck der Arbeit 24

14 Zusammenfassung und Ausblick 25

Literaturverzeichnis 26

Anhang 27

Eidesstattliche Versicherung 28

Auszug aus dem Strafgesetzbuch (StGB) 28

15 Publication bibliography 30

# List of Figures

[Fig. ‎1.1: AI methods evaluation 2](#_Toc145963398)

[Fig. ‎2.1: Essential element in Smart Standard 5](https://uniduede.sharepoint.com/sites/SMARTStandards-AustauschStudierende2-MAMohamedAlazzazy/Freigegebene%20Dokumente/MA%20Mohamed%20Alazzazy/Mohamed_Unterlagen/Unterlagen/Abschlussarbeitvorlage_Alazzazy.docx#_Toc145963399)

[Fig. ‎2.2: Strukturelement in Smart Standard 5](#_Toc145963400)

[Fig. ‎2.3: Differentiation elements in Smart Standard 5](#_Toc145963401)

[Fig. ‎2.4 Text extraction pipeline 7](#_Toc145963402)

[Fig. ‎3.1: Grid Search (Rosebrock 2021) 13](#_Toc145963403)

[Fig. ‎3.2 : Grid Search (a) and Random Seach (b) 13](#_Toc145963404)

[Fig. ‎3.3: Bayesian adaptation 13](#_Toc145963405)

[Fig. ‎3.4: Genetic algorithms 13](#_Toc145963406)

# List of Tables

[**Tab. 3.1:** Quellenangaben bei Abbildungen 8](#_Toc142140116)

[**Tab. 3.2:** Übersicht der Formatempfehlungen für Abbildungen und Tabellen 9](#_Toc142140117)

# List of Abbreviations

AI Artificial Intelligence

IoT Internet of Things

ISO International Organization for Standardization

NISO National Information Standards Organization

STS Standard Tag Suite

DIN German Institute for Standardization

PDF Portable Document Format

XML Extensible Markup Language

NLP Natural Language Processing

IE Information extraction

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

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

NN Neural Network

CNN Convolutional Neural Network

SVM Support Vector Machine

GS Grid Search

RGS Random Grid Search

# Symbolverzeichnis

|  |  |
| --- | --- |
|  | Anschaffungsauszahlung in |
|  | Kapitalwert |
|  | Einzahlungsüberschuss in Bezug auf |
|  | Kalkulationszinsfuß |
|  | Nutzungsdauer |
|  | Zinsfaktor |
|  | Abstand der Stufe s in cm vom Seitenrand |
|  | Stufenindex |
|  | Periodenindex |

Anmerkung

Ein Symbolverzeichnis wird nicht notwendigerweise in jeder Arbeit benötigt.

# Introduction

A dynamic collaboration of trends and challenges is developing in the field of modern product development. This dynamic is driven by the convergence of different national and sociocultural frameworks and the evolving landscape of product service systems, resulting in a plethora of customer requirements (Bertoni 2018).

At the center of this varied environment are the smart standards that provide prescriptive descriptions of the requirements for products, services, and processes. These documents have a significant impact on key design decisions. However, the widespread print or PDF format in which they are consumed requires complicated and often exhausting manual information extraction processes, underscoring the need to automate this important process. Thus, the current usage of standards is inefficient and not compatible with digital knowledge management and its integration into product development processes (din-en 2023)(Luttmer et al. 2023).

Integrating automation into these processes promises multiple benefits. These include speeding up the preparation of quotations, refined costing, adapting products more quickly to customer requirements, rigorous quality control by reducing errors, and ultimately reduced time to market In the pursuit of digitization, initiatives such as the central XML database of the German Institute for Standardization (DIN) and the development of the schema NISO STS (Standard Tag Suite) by the National Information Standards Organization (NISO) have played a central role (Rigger and Vosgien 2018) (Luttmer et al. 2023).

This evolving industrial landscape is being profoundly shaped by transformative technologies such as the Internet of Things (IoT), Big Data analytics, robotics, and artificial intelligence (AI), which together are creating a highly connected world. In particular, the strategic use of AI, especially in the digitization of requirements and standards, streamlines operational processes and optimizes resource allocation (Ioannis 2019) (ISO 2023).

Machine learning (ML), as a central subfield of AI, represents a cornerstone of the modern technological paradigm. It focuses on algorithmic analysis of datasets and generation of predictive insights (Dunjko and Briegel 2017). Considerable progress has been made in the field of document analysis, especially through natural language processing and machine learning techniques (Kili Technology 2022).

Despite the inherent potential, a significant portion of companies remain reluctant to embrace automation. Remarkably, 75% of companies view the importance of AI and similar approaches as low (Giering 2022).

This striking discrepancy between potential and adoption underscores the compelling importance of the present research.

## Problem Description

In the age of digital transformation, the digitization of standards aims to improve accessibility and usability. But despite significant progress, a major challenge remains.

First, standards were converted from paper to digital formats such as PDF, which was a remarkable advance. Then, these digital documents were further transformed into structured, machine-readable formats, primarily using XML (Luttmer et al. 2021). But the digitalization of standards, from paper to PDF to structured XML, is not a sufficient way to ensure cross-domain interoperability in an era of digital transformation. There is an increasing need for intelligent, machine-executable data in standards (LOIBL et al. 2020).

Farthermore, Information extraction (IE) aims to capture and extract useful information from unstructured or semi-structured data and the requirement is an important factor of these information. The extraction of requirements from engineering documents is a special application of IE which has not been investigated in detail in the standardization domain (Luttmer et al. 2023).

Apart from standards, the identification and extraction of requirements has been examined by different researchers with respect to many types of textual data, e.g. specifications [12-15], or engineering documentation in general [17]

In order to facilitate the use of standards, the information therein and, in particular, requirements – as main component of standards – need to be automatically extracted from standards documents and transferred to, e.g., requirements management tools. As shown in section 1.2, different techniques are used to extract requirements from various document types.

Therefore there are several advanced developments to integrate the artificial intelligence (AI) in standard digitization process, in order to extract the requirement from norm standards formed in PDF automatically, but how to identify the most suitable ML algorithm for potential applications to solve prevailing problems is now the quistion. As undertaken by our research team in the development of "Smart Standards," has revealed some shortcomings. While AI-driven methods have demonstrated promise, achieving the desired level of accuracy remains a formidable hurdle.

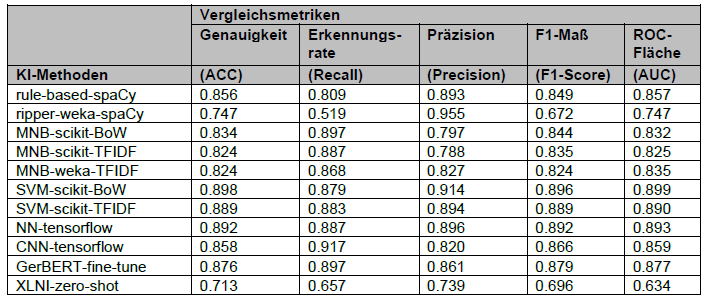


Fig. ‎1.1: AI methods evaluation (Vitalijs Prihodko 2022)

Our research findings indicate that, despite the implementation of AI technologies in standard digitization, the accuracy levels as it showen in Figure1.1 attained hover at approximately 89% of requirement extraction. While this represents a commendable achievement, the 11% margin of inaccuracy has tangible repercussions. These inaccuracies translate into inefficiencies in time, financial resources, and effort required to rectify discrepancies.

## Goal of the Thesis

In order to automatically integrate digital standards into development processes and tools, standard

content needs to be offered in machine-actionable format (Loibl et al., 2020). As this requires the

representation of semantic relationships, which cannot be achieved by the currently used XML format,

content needs to be transferred to an appropriate representation type.

This master's thesis endeavors to investigate the challenge of analyzing and optimizing suitable differentiated artificial intelligence methods from existing work in order to achieve optimal accuracy in requirements and norms identification. Therefore, achiving a step in digitization of standards through contributing to the advancement of Smart Standards and the broader field of AI-driven standardization.

As shown in section 1.1 knowledge graphs are used to provide semantic enriched content. This

research paper aims to

## Structure of thesis

The thesis is organized into eight chapters, each contributing to a comprehensive exploration of the topic. Chapter 1, serving as the introduction, lays the foundation by presenting the problem description and the overarching goal of the thesis. In Chapter 2, the theoretical foundation is constructed, covering the concepts of Smart Standards and the integration of text classification with AI. Chapter 3 delves into the state of the art, shedding light on the intricacies of parameters and hyperparameters tuning in the context of AI-driven standard digitization. Building upon this, Chapter 4 elucidates the methodology, elucidating the research design, data collection, and AI model development processes.

Chapter 5, the analysis chapter, critically examines the results obtained from AI-driven standard digitization experiments, providing insights into accuracy levels and performance. Following this, Chapter 6 takes the lead in concept development, proposing innovative solutions to enhance accuracy and address challenges identified in the research. Chapter 7, the development process, provides a comprehensive account of the practical implementation of proposed solutions, accompanied by rigorous testing and validation.

Finally, Chapter 8 serves as the conclusion, summarizing the key findings and conclusions drawn from the research. In addition to offering a concise recap, this chapter explores potential future research directions, charting the course for subsequent investigations based on the valuable insights derived from this thesis.

# Theoretical foundation

In this chapter Smart standard, Text extraction and Artificial Intelligence (Vitalijs Prihodko 2022)

## Smart standard

Standards play a vital role as documents that outline requirements for products, services, and processes, facilitating the smooth flow of goods, promoting exports, and ensuring efficiency, quality, and safety. Traditionally, standards were available in paper form, and more recently, they have become accessible in PDF format due to technological advancements.

While paper documents require manual searching and processing, PDFs offer a slightly faster method through full-text search. However, even with PDFs, users still need to search for specific information and manually process it, either individually or with the help of software, before integrating the findings into their processes.

For instance, users may need to search for mathematical formulas or tolerance limits of screw threads, but this highlights a limitation in the current format. PDFs do not allow for automatic processing of information beyond simple search functionalities. As a result, further automation and integration of data from standards require advanced techniques beyond what PDFs can provide.

The aim of smart standards is to be available to industry in such a way that, ideally, they would automatically fit into the control loop of production, for example, and could flow directly into the industry's digital systems, concepts, and formats. Adjustments and evaluations could then be adopted in a targeted manner in the various systems. A clear time advantage with considerable cost savings for the industry.

In addition, digitalization of standards’ formats like PDF turned out to be very challenging as they are very optimized for readability by the human eye. So, a new way must be found to make the documents easier for humans and machine to access and extract information. In other words, the goal here is to automate the process of extracting information from standards.

### Core elements in smart standards

Each standards document consists of various components, some of which are obligatory and others optional. The following lines briefly discuss the various elements of the standards document and distinguish them according to these criteria:

* Normative element - element describes the scope of a document or contains specifications.
* Informative element - element intended to facilitate the understanding and applicability of the document or to provide contextual information about the content, background, or relationship to other documents.
* Unconditionally required element - element that is committed to its existence in a document.
* Optional element - element whose existence depends on the specifics of the document in question.
* Optional element - element that the author of a document may choose to include or not.

A close-up of a list

Description automatically generatedA screenshot of a computer

Description automatically generated

Fig. ‎2.1: Essential element in Smart Standard

Fig. .: Strukturelement in Smart Standard

A close-up of a document

Description automatically generated

Fig. .: Differentiation elements in Smart Standard

1. **Text components: -**
2. Sections and subsections
3. Enumerations
4. Notes
5. Examples
6. Footnotes
7. Mathematical equations
8. Images
9. Tables
10. **Specifications in normative documents**: -.
    1. **requirement** - This refers to a requirement in the content of a document that expresses objectively verifiable criteria that must be met without any deviation if compliance with the document is required. For requirements that refer to range limits or boundary conditions, the following expressions are used:

* „darf nur“- for mandatory requirements.
* „darf höchstens“- for maximum requirements.
* „muss mindestens“- for minimum requirements.
* When such requirements refer to points in time, the following expressions shall be used:
* „muss spätestens“- for the latest permissible time.
* „darf frühestens“- for the earliest permissible time. (Vitalijs Prihodko 2022)

### Information extraction

Smart standards often deal with vast amounts of data. Information Extraction is employed to parse and extract valuable information from these data sources, converting them into a structured format that can be easily processed and analyzed by smart systems and applications.

* Named Entity Recognition (NER): NER is a fundamental IE technique that identifies and classifies entities such as names of persons, organizations, locations, dates, and numerical values in unstructured text. In the context of smart standards, NER can be used to identify specific entities related to domain-specific terms, sensor measurements, or equipment names.
* Relation Extraction: This technique focuses on identifying the relationships between entities mentioned in the text. In the context of smart standards, relation extraction can be used to identify associations between devices, processes, or standards, enabling better understanding and interoperability. As in our group with master thesis “Development of automation approaches for the extraction and modeling of relationships between formulas in standard documents”.
* Event Extraction: Event extraction involves detecting and extracting events or actions from text data. For example, in the domain of smart standards, event extraction can be used to identify changes in standard versions, updates, or new protocols being introduced.
* Text Classification: Text classification is used to categorize unstructured text into predefined classes or categories. In smart standards, text classification can be applied to automatically classify documents or data into specific standards, protocols, or use cases. And that is our focus.

## Artificial Intelligence and Text extraction

### Artificial Intelligence techniques

### Text Extraction

**Data Preprocessing**

**Feature Extraction**

**Modeling**

**Evaluation**

**Deployment**

Fig. ‎2.4 Text extraction pipeline

Text extraction typically involves processing unstructured or semi-structured text data to extract relevant information, such as entities, keywords, or sentiment, from the text. In Python, the pipeline for text extraction typically involves the following steps:

1. Data Preprocessing: This step involves cleaning and preparing the raw text data for further analysis. This can involve tasks such as removing punctuation, lowercasing text, removing stop words, and tokenizing the text into individual words.

• Remove any unwanted characters: You can use regular expressions to remove unwanted characters such as punctuation marks, special characters, and numbers from the text data.

• Tokenization: Tokenization is the process of breaking text data into individual words or tokens. You can use the NLTK library in Python to tokenize German text data.

• Remove stop words: Stop words are common words that do not provide any value to the analysis, such as "the", "and", "or", etc. You can remove stop words using the NLTK library.

• Stemming or Lemmatization: Stemming is the process of reducing words to their root form by removing suffixes, whereas lemmatization involves reducing words to their base form. Both techniques can be used to reduce the number of unique words in the text data.

• Text normalization: This step involves converting all text to lowercase to ensure that similar words are treated equally.

• Removing duplicates: You can remove any duplicates that may be present in the text data.

• Encoding: Finally, you can encode the cleaned text data in a format that can be used for analysis, such as a bag-of-words model or a TF-IDF matrix.

2. Feature Extraction: Once the text data has been preprocessed, the next step is to extract features from the text that can be used for analysis. Common techniques for feature extraction include bag-of-words models, TF-IDF (term frequency-inverse document frequency) weighting, and word embeddings.

3. Modeling: With the features extracted, the next step is to train a model that can perform the specific text extraction task. This can involve supervised learning techniques such as classification or regression, or unsupervised learning techniques such as clustering or topic modeling.

4. Evaluation: Once the model has been trained, it is important to evaluate its performance on a validation dataset. This can help identify any issues with the model, such as overfitting or underfitting, and can guide further refinement of the model.

5. Deployment: Finally, the trained model can be deployed for use in production, where it can be used to extract information from new text data.

### Text extraction techniques

1. NLTK: -

The Natural Language Toolkit (NLTK) is a popular Python library used for natural language processing tasks such as text extraction, tokenization, part-of-speech tagging, and more. While NLTK is a powerful tool for text extraction.

2. Spacy: -

SpaCy is another popular Python library used for natural language processing tasks such as text extraction, named entity recognition, dependency parsing, and more.

3. TextBlob: -

TextBlob is a Python library that provides a simple interface for common natural language processing tasks such as sentiment analysis, part-of-speech tagging, and noun phrase extraction.

4. Stanford CoreNLP: -

Stanford CoreNLP is a Java-based natural language processing toolkit that provides a range of tools for text processing, including tokenization, part-of-speech tagging, named entity recognition, and dependency parsing.

5. Gensim: -

Gensim is a Python library for topic modeling, document similarity, and natural language processing tasks such as text extraction.

6. Pattern: -

Pattern is a Python library for web mining, natural language processing, machine learning, and network analysis.

7. Scikit-learn: -

Scikit-learn is a Python library for machine learning and data mining that provides a range of tools for text extraction and processing.

8. PyTorch: -

PyTorch is a popular open-source machine learning library developed by Facebook. It provides a wide range of tools and functionality for building deep learning models, including models for text extraction in Python.

# State of the art

In this chapter Parameters and hyperparameters and Hyperparameters tuning.

## Parameters and Hyperparameters

In machine learning, we need to differentiate between model parameters and model hyperparameters:

1. Model Parameters: they are the internal variables or coefficients that a machine learning model learns or estimates from the given dataset during the training process. These parameters are integral to the model's functioning, as they capture the underlying patterns, relationships, and representations within the data. The specific values of these parameters significantly affect the model's predictive power and its ability to generalize to new, unseen data. For example the weights of a deep neural network. (Anyscale 2023)
2. Hyperparameters: on the other hand, they are specific to the algorithm itself, so we can’t calculate their values from the data. We use hyperparameters to calculate the model parameters. Different hyperparameter values produce different model parameter values for a given data set. For example, the learning rate in deep neural networks. (Anyscale 2023)

(Anyscale 2023)

|  |  |
| --- | --- |
| Parameters | Hyperparameters |
| They are required foe making predictions. | They are required for estimating the model parameters. |
| They are estimated by optimization algorithms(Gradient Descent, Adam, Adagrad). | They are estimated by hyperparameter tuning. |
| They are not set manually | They are set manually |
| The final parameters found after training will perform on unseen data. | The choice of hyperparameters decide how efficient the training is. In gradient descent the learning rate decide how efficient and accurate the optimization process in estimating the parameters. |

### Neural Network (NN) and Convolutional (CNN)

Hyperparameters in a neural network are the settings and configurations that are chosen before the training process begins. They are not learned during training but rather set by the developer or researcher. Hyperparameters have a significant impact on the network's performance, generalization, and training speed. Some common hyperparameters in a neural network include:

1. Learning rate: The step size at which the optimizer updates the weights during training.

2. Number of hidden layers: The depth of the neural network, i.e., the number of layers between the input and output layers.

3. Number of units/neurons in each hidden layer: The number of nodes in each hidden layer.

4. Activation functions: The non-linear functions applied to the output of each neuron.

5. Batch size: The number of samples used in one forward/backward pass during training.

6. Number of epochs: The number of times the entire dataset is passed through the network during training.

7. Regularization parameters: Parameters used to apply regularization techniques like L1 or L2 regularization to prevent overfitting.

8. Optimizer: The optimization algorithm used to update the weights, such as SGD, Adam, RMSprop, etc.

9. Dropout rate: The probability of dropping out a neuron during training to reduce overfitting.

10. Loss function: The function used to measure the difference between predicted and actual values during training.

11. Initialization of weights: The method used to initialize the weights of the neural network.

12. Activation function for the output layer: The activation function used for the output layer is based on the type of task (e.g., softmax for classification, linear for regression).

### Support Vector Machines (SVM)

The Support Vector Machine (SVM) algorithm has two sets of hyperparameters, classification and vectorization, that can be tuned to optimize its performance on a given data set.

Classification

For this set there are some hyperparameters:

1. C (Regularization Parameter): The C parameter controls the trade-off between achieving a low training error and a low testing error. Smaller values of C lead to a wider margin but more margin violations, which may lead to better generalization on the test data. Larger values of C result in a narrower margin and fewer margin violations, which may lead to better fitting to the training data but potentially overfitting.
2. Max features: set an upper limit on the number of unique words (features) that the Count Vectorizer will consider. If the vocabulary of the input data contains more words than this specified limit, the Count Vectorizer will keep only the most frequent max features words based on their frequency of occurrence in the entire dataset. This can be useful for reducing the dimensionality of the dataset and improving model training efficiency.
3. Class Weights: SVM can handle imbalanced datasets using class weights. The class weight parameter allows you to assign different weights to different classes to give more importance to the minority class.
4. Kernel: SVM can use different types of kernels, such as linear, polynomial, radial basis function (RBF), and others. The choice of the kernel depends on the nature of the data and the problem. For example, RBF (Gaussian) kernel is often suitable for non-linearly separable data.
5. Gamma: is a parameter for 'poly', 'rbf', and 'sigmoid' kernels. It defines how far the influence of a single training example reaches. Low values mean 'far,' and high values mean 'close.' It can be set to 'scale' or 'auto' for automatic calculation based on the dataset.
6. Coef0: It's the independent term in the kernel function. It is only significant in 'poly' and 'sigmoid' kernels. It can affect the shape of the decision boundary.
7. Degree: This parameter is used with polynomial kernels (`'poly'`). It specifies the degree of the polynomial kernel function. Higher degrees can capture more complex relationships but may lead to overfitting.
8. Shrinking: This parameter controls whether to use the shrinking heuristic. If set to `True`, it enables the shrinking optimization, which may speed up training but may also affect the model's accuracy slightly.
9. Probability: Setting this to `True` enables probability estimates during training. It can be useful if you want to obtain class probabilities alongside predictions.
10. Tol: It's the tolerance for stopping criterion. The algorithm stops training when the change in the objective function (duality gap) is smaller than this value.

Vectorization

For this type there are some hyperparameters:

1. TF-IDF (Term Frequency-Inverse Document Frequency): is a technique used to convert text data into numerical form. It evaluates the importance of a word in a document relative to a collection of documents (corpus). Words that appear frequently in the document but not in many others are considered more important. TF-IDF assigns a weight to each word based on its frequency and rarity.
2. Count Vectorization: is a simple technique to convert text data into numerical format. It creates a matrix where each row represents a document, and each column represents a unique word in the entire corpus. The values in the matrix indicate how many times each word appears in each document.
3. Word Embeddings: are dense vector representations of words in a high-dimensional space. They capture semantic relationships between words based on their context in a large text corpus. Word embeddings like Word2Vec or GloVe allow machines to understand the meaning and similarity between words, making them useful for natural language processing tasks.

## Hyperparameter tuning techniques

Hyperparameter tuning is a crucial step in machine learning model development, as it helps you find the best set of hyperparameters for the model. while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model’s performance, minimizing a predefined loss function to produce better results with fewer errors. Note that the learning algorithm optimizes the loss based on the input data and tries to find an optimal solution within the given setting. However, hyperparameters describe this setting exactly (Nabi 2019) (Anyscale 2023) .

There are some techniques for hyperparameters tuning will be discussed:

Manual trial-and-error Tuning, Grid Search, Random Search, Bayesian Optimization, Genetic Algorithms, Gradient-Based Optimization, Ensemble Methods, Automated Machine Learning (AutoML) and Cross-Validation.

### Manual trial-and-error Tuning

With manual tuning, based on the current choice of parameters and their score, we change a part of them, train the model again, and check the difference in the score, without the use of automation in the selection of parameters to change and value of new parameters.

For Manual tuning offers both advantages and disadvantages, making it a valuable approach in certain scenarios while challenging in others:

Advantages: Manual tuning allows users to develop a deep understanding of hyperparameter behavior so that they can leverage this knowledge in future projects, increasing the efficiency of hyperparameter tuning in subsequent projects. Users can apply their domain knowledge and problem-specific insights to make intuitive changes to hyperparameters, improving model performance by incorporating context. This technique provides the flexibility to quickly adapt to new challenges or evolving data distributions, making it particularly valuable in dynamic environments. In addition, manual tuning is computationally less demanding, resulting in cost-effective use of computational resources, making it a budget-friendly option for projects with limited resources. It excels at the fine-tuning stage, allowing users to further refine an already good model and achieve optimal performance with minimal adjustments.

Disadvantages: However, manual tuning of hyperparameters also brings challenges. It requires a significant manual effort that can be resource intensive and may not be feasible for large projects. The process can be very time-consuming, requiring numerous trial-and-error iterations to determine optimal hyperparameters, which can slow down model development. Tuning depends on the user's intuition, which makes it subjective and prone to variations in parameter selection by different people. Manual tuning may not fully explore the entire hyperparameter space, so the absolute best hyperparameter settings may not be found. Exact replication of the tuning process can be difficult and lead to inconsistencies when others attempt to replicate the results. Excessive adjustments during manual tuning can lead to overfitting, where the model performs well on training data but performs poorly on new, unknown data.

In summary, manual tuning of hyperparameters offers valuable advantages, such as deep understanding, flexibility, and cost effectiveness, but comes with the disadvantages of subjectivity, time-consumption, and potential difficulty in reproducing results. Careful consideration of project requirements and constraints is essential when deciding whether to use manual hyperparameter tuning in a machine learning project.

### Grid Search

Grid search involves defining a range of possible values for each hyperparameter. It exhaustively tries all combinations of hyperparameters and evaluates the model's performance using cross-validation as shwed in Figur 3.1.

A diagram of a grid search

Description automatically generated

Fig. .: Grid Search (Rosebrock 2021)

Grid search has certain advantages and disadvantages that affect its applicability.

Advantages: Systematic exploration within a predefined grid leaves no room for missing optimal configurations. It eliminates manual guesswork and intuition and provides an objective approach. The results obtained with Grid Search are highly reproducible, which increases the reliability of experiments. Moreover, it is easy to implement, making it accessible to novices.

Disadvantages: However, Grid Search can be computationally intensive, especially in scenarios with numerous hyperparameters or large ranges of values. As the size of the hyperparameters increases, the size of the grid grows exponentially, making Grid Search impractical for high-dimensional spaces. There is a risk that optimal hyperparameters outside the predefined grid will not be found. The implementation of Grid Search requires significant computational resources and is not adaptive compared to advanced techniques such as Bayesian optimization.

In summary, Grid Search provides a systematic and reproducible approach for small hyperparameter spaces and novices, but may be impractical for high-dimensional tasks due to its high computational cost and limited adaptability. Careful consideration is required when selecting Grid Search as a hyperparameter optimization method.

### Random Search

Random search randomly samples hyperparameters from predefined ranges. It's less computationally expensive than grid search but often performs just as well or better, especially when some hyperparameters matter more than others.

We can control the randomness by assigning density function of parameters instead of specific value, e.g. uniform distribution or normal distribution.

Random search offers several advantages and disadvantages that affect its suitability for machine learning tasks.

Advantages:

Random search is computationally more efficient than exhaustive methods such as grid search, especially when navigating extensive hyperparameter spaces, since it does not require testing every possible combination. It has a high probability of finding near-optimal hyperparameters because it randomly examines different combinations. The implementation of Random Search is relatively simple, making it a pragmatic choice for initial hyperparameter tuning. In addition, Random Search is resource efficient and requires fewer computational resources compared to exhaustive search methods.

Disadvantages:

Despite its efficiency, random search does not guarantee that the absolute best hyperparameters will be found, and multiple iterations may be required to converge to optimal values. The inherent randomness in hyperparameter selection can lead to variation in results between runs, which can undermine reproducibility. While random search is less resource intensive than grid search, it can still be computationally intensive, especially for complex models or large datasets. In cases where the optimal hyperparameters are at the extremes of the search space, it may take longer to find them using Random Search. In addition, Random Search is not very adaptive since it does not adjust its sampling based on previous iterations, making it less efficient than adaptive methods such as Bayesian optimization.

In summary, Random Search is computationally efficient and provides a reasonable chance of finding good hyperparameters, making it suitable for initial hyperparameter tuning. However, it does not provide a guarantee of optimality, can lead to variability, and can be resource intensive in certain scenarios. Therefore, careful consideration of the associated drawbacks is required when selecting random search as a method for hyperparameter optimization.

A diagram of a model accuracy

Description automatically generated

Fig. .2 : Grid Search (a) and Random Seach (b)

Depending on the number of searches and how large the parameter space is, some parameters might not be explored enough. RGS allows the user to better control the number of optimization rounds needed by adapting this number to the available computational resources. It has been shown that, in general, RGS tends to provide better and faster results than GS because it is more efficient in the exploration of the hyperparameter space.

### Bayesian Optimization

Bayesian optimization uses probabilistic models to model the relationship between hyperparameters and model performance. It selects new hyperparameters to try based on the model's predictions, with the goal of finding the optimal values quickly. This method is efficient for complex and computationally expensive models.

It starts from random and narrowing the search space based on Bayesian approach as it showen in the next figure.A group of graphs showing different functions

Description automatically generated

Fig. .: Bayesian adaptation

Bayesian optimization emerges as an effective method with distinct advantages and dis advantages.

Advantages: Its exceptional efficiency in identifying optimal hyperparameters, especially for computationally intensive objective functions, underscores its practicality. Moreover, the ability to account for noisy objective functions enhances the robustness of the method in real-world applications. By cleverly balancing exploration and exploitation, Bayesian optimization systematically finds optimal solutions in complex hyperparameter spaces, which makes it particularly valuable for global optimization tasks.

Disadvantages: However, the implementation complexity of the method surpasses simpler techniques such as grid search or random search, so a deeper understanding of its mechanisms is required. The selection of hyperparameters for the Gaussian process model, while critical, can be complicated and requires expertise. Despite its efficiency in function evaluation, Bayesian optimization can incur computational costs, especially for large data sets or complex models, potentially limiting scalability. Its primary applicability to continuous hyperparameter spaces may limit its use in discrete or combinatorial problems. Sensitivity to initialization underscores the need for careful configuration choices.

In summary, Bayesian optimization provides an efficient and robust approach to hyperparameter tuning that is ideal for costly evaluations and extensive parameter exploration. However, the complexity of the method, the difficulty in selecting hyperparameters, the computational cost, the space limitations, and the sensitivity of initialization warrant careful evaluation of its use for hyperparameter tuning in machine learning research and practical applications.. (Anyscale 2023)

### Genetic Algorithms

Genetic algorithms are inspired by the process of natural selection. They create a population of sets of hyperparameters, evaluate their performance, and then evolve the population over several generations to find optimal or near optimal hyperparameters.

* Population: Population is the subset of all possible or probable solutions, which can solve the given problem.
* Chromosomes: A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome.
* Gene: A chromosome is divided into a different gene, or it is an element of the chromosome.
* Allele: Allele is the value provided to the gene within a particular chromosome.
* Fitness Function: The fitness function is used to determine the individual's fitness level in the population. It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function.
* Genetic Operators: In a genetic algorithm, the best individual mate to regenerate offspring better than parents. Here genetic operators play a role in changing the genetic composition of the next generation.
* Selection

The algorithm begins by creating a random initial population.

A diagram of a process

Description automatically generated

Fig. ‎3.4: Genetic algorithms

Figure 7

Start: It generates a random population of n chromosomes.

Fitness: It calculates the fitness f(x) of each chromosome x in the population.

Selection: It chooses two parent chromosomes from a population as per their fitness. The better fitness, the higher the probability of getting selected.

Crossover: In crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, the offspring is the exact copy of the parents.

Mutation: In mutation probability, mutate new offspring at each locus.

Accepting: It places new offspring in the new population.

Replace: It uses the newly generated population for a further run of the algorithm.

1. Advantages:
   1. The parallel capabilities of genetic algorithms are best.
   2. It helps in optimizing various problems such as discrete functions, multi-objective problems, and continuous functions.
   3. It provides a solution for a problem that improves over time.
   4. A genetic algorithm does not need derivative information.
   5. It can handle discrete and continuous hyperparameters.
   6. It can find unconventional hyperparameter combinations.
2. Disadvantags:
   1. Computationally expensive due to multiple evaluations of fitness.
   2. Requires careful design of genetic operators (crossover, mutation, etc.).
   3. Genetic algorithms are not efficient algorithms for solving simple problems.
   4. It does not guarantee the quality of the final solution to a problem.
   5. Repetitive calculation of fitness values may generate some computational challenges.

(www.javatpoint.com 2023)

### Gradient-Based Optimization

In the gradient based algorithm, we start with random model parameters and calculate the error for each learning iteration, keep updating the model parameters to move closer to the values that results in minimum cost. Please refer my post for details. Gradient descent algorithms multiply the gradient (slope) by a scalar known as the learning rate (or step size) to determine the next point. This parameter tells how far to move the weights in the direction of the gradient. It relies on the concept of gradients, which are vectors that point in the direction of the steepest increase in a function. In the context of machine learning, this function is typically a cost or loss function that measures the error between predicted and actual values. Some libraries like Keras Tuner and Optuna provide gradient-based optimization methods. They adaptively update hyperparameters based on gradients, similar to how gradient descent optimizes model parameters.

A diagram of a weight loss

Description automatically generated

Figure 8 (Nabi 2019)

1. Advantages:
   1. Efficiency: It's fast and can handle large amounts of data.
   2. Versatility: Works with many machine learning methods.
   3. Local Minimum Search: Good at finding good solutions in most cases.
   4. Ease of Use: It's relatively straightforward to implement.
2. Disadvantages:
   1. Learning Rate Challenge: Picking the right "learning rate" is tricky.
   2. Local Minimum: Can sometimes get stuck in okay solutions(local minimum).
   3. Saddle Points: Slow to escape tricky points in the optimization landscape.
   4. Smoothness Needed: Works best with smooth functions.
   5. Initialization Matters: Starting point can affect results a lot.
   6. High Dimensions: Can struggle in very complex problems, especially in deep learning.
   7. Resource Demands: Requires a lot of memory and computation for deep learning.

Despite these challenges, gradient-based optimization is still widely used and effective, especially with advanced techniques to overcome its limitations.

### Ensemble Methods

You can use ensemble methods like stacking to combine the predictions of multiple models with different sets of hyperparameters. This can often lead to better performance than tuning a single model. The idea behind ensemble methods is that by combining the strengths of multiple models, you can often achieve better results than using a single model.

In such ensembles, predictions from one machine learning model become predictors for another (next level). The figure below shows some variations of ensembles where the data is transferred from left to right.

A diagram of different forest types

Description automatically generated

Figure

1. Advantages: -
   1. Improved Performance: Ensemble methods often result in better predictive performance compared to individual models. They can reduce overfitting and improve generalization.
   2. Robustness: Ensembles are less sensitive to noise in the data and can handle outliers and errors more effectively.
   3. Versatility: Ensemble methods can be applied to a wide range of machine learning algorithms, making them versatile for different types of problems.
   4. Model Agnostic: They are model agnostic, meaning they can work with any base model, allowing you to choose the best model for your specific problem.
   5. Interpretability: Some ensemble methods, like Random Forests, provide feature importance scores, which can help interpret the importance of different features in the data.
2. Disadvantages: -
   1. Complexity: Ensemble methods can be more complex to implement and tune compared to individual models. They often require more computational resources.
   2. Overfitting: While ensembles can reduce overfitting, they can still overfit if not properly tuned or if the base models are overfit themselves.
   3. Interpretability: In some cases, the final ensemble model may be less interpretable than the individual base models, especially when using complex ensemble methods.
   4. Computational Cost: Training and using ensembles can be computationally expensive, especially when combining a large number of base models.
   5. Lack of Diversity: Ensembles perform best when the base models are diverse and make different types of errors. If base models are too similar, the ensemble may not provide significant improvements.

### Automated Machine Learning (AutoML)

Different ML models are appropriate for different applications. In order to find the most suitable algorithm, the simple method of applying and optimizing all known learning algorithms is not practical in most cases. This process of finding the best ML algorithm and creating the optimal architecture by setting the best hyperparameters is a complex and time-consuming process. Here comes the importance of an Automated Machine Learning (AutoML) system that determines the optimal configurations for a particular application with the best performance within the time constraints. For a deep learning network, AutoML not only performs Hyperparameter Optimization (HPO) to automatically set the optimal hyperparameters but also selects the right neural architecture for each layer. It also provides tools and approaches for enabling ML to be accessible to non-experts, increasing performance, and speeding up ML research.

1. Advant age: -
   1. Accessibility: AutoML lowers the entry barrier to machine learning, allowing individuals with limited expertise to leverage powerful machine learning techniques.
   2. Time Efficiency: It automates time-consuming tasks like feature engineering, hyperparameter tuning, and model selection, significantly reducing the time required to build and deploy models.
   3. Reduced Human Error: Automation reduces the chances of human errors in the model development process, leading to more robust and reliable models.
   4. Scalability: AutoML can be scaled to handle large datasets and complex model architectures, making it suitable for various applications.
   5. Consistency: AutoML tools provide consistent and reproducible results, which is essential for research, development, and compliance purposes.
   6. Democratization: It democratizes machine learning by enabling domain experts, data analysts, and non-experts to leverage machine learning for their specific use cases.
   7. Hyperparameter Optimization: AutoML tools often incorporate sophisticated hyperparameter optimization techniques to fine-tune models effectively.
2. Disadvantages: -
   1. Loss of Control: Users may have less control over the fine-grained details of model development, which can be a limitation for experts who require specific configurations.
   2. Black Box Models: Some AutoML solutions generate complex models that are challenging to interpret, potentially reducing transparency and trust in the model's predictions.
   3. Limited Customization: AutoML tools might not support custom model architectures or specialized algorithms, limiting their applicability for certain tasks.
   4. Resource Intensive: Depending on the complexity of the problem and the size of the dataset, AutoML can still be resource-intensive in terms of computational power and memory.
   5. Cost:Some commercial AutoML platforms may come with subscription fees or usage charges, which can be costly for organizations with large-scale machine learning needs.
   6. Not Always Optimal: While AutoML can generate high-performing models, it may not always find the absolute best model for a particular problem. Manual fine-tuning by experts may still be necessary for certain tasks.
   7. Data Dependency: The effectiveness of AutoML depends on the quality and quantity of the input data. If the data is inadequate or noisy, AutoML might not perform well.

(Vincent and Jidesh 2023)

### Cross-Validation

Assess the performance of a predictive model and to mitigate issues related to overfitting and data variability. It involves partitioning the available dataset into multiple subsets, training the model on some of these subsets, and evaluating its performance on the remaining subsets. Cross-validation provides a more robust estimate of a model's performance compared to a single train-test split.Always use cross-validation to evaluate the performance of different hyperparameter settings. This helps ensure that your hyperparameter choices generalize well to unseen data.

A black background with purple rectangles

Description automatically generated

Fig. ‎3.5: Cross validation diagram (scikit-learn 2023)

1. Advantages: -
   1. Robust Performance Estimate: Cross-validation provides a more robust estimate of a model's performance compared to a single train-test split. It reduces the impact of data variability on performance assessment.
   2. Effective for Small Datasets: It is particularly useful when the dataset is small because it maximizes the use of available data for both training and testing.
   3. Bias Reduction: Cross-validation helps in reducing bias in performance estimation, especially when dealing with imbalanced datasets or when a single random split might be unrepresentative.
   4. Model Selection: It assists in selecting the best model or hyperparameters by comparing the performance of different configurations across multiple runs.
   5. Diagnostic Tool: It can be used as a diagnostic tool to identify issues like overfitting or underfitting, as performance metrics are computed for different data subsets.
2. Desadvantages: -
   1. Computational Cost: Cross-validation can be computationally expensive, especially for large datasets and complex models, as it involves multiple training and testing iterations.
   2. Complexity: Implementing cross-validation requires more coding effort compared to a single train-test split.
   3. Data Leakage: If not used carefully, cross-validation can lead to data leakage if preprocessing steps (e.g., scaling) are applied to the entire dataset before splitting.
   4. Resource Intensive: It may require a significant amount of memory and computing power, which can be a limitation in resource-constrained environments.
   5. Randomness: The choice of 'k' and the randomness in data splitting can affect cross-validation results, although this is usually mitigated by taking the average over multiple runs.
   6. Difficulty in Communication: Domain experts may struggle to communicate complex concepts to individuals outside their field, leading to misunderstandings or miscommunications.
   7. Limited Perspective: Deep domain knowledge can sometimes lead to a limited perspective, where experts may overlook broader societal or ethical implications.
   8. Dependency: Overreliance on domain experts can create dependencies within an organization, making it challenging to adapt to personnel changes.
   9. Competitive Disruption: In rapidly evolving fields, domain knowledge that is too entrenched in the status quo may be disrupted by newcomers with fresh perspectives.

We can use a compination of those techniques to get the best behaviour ??

# Methodology

# Analysis

# Concept development

## Initial combination: -

## Hyperparameters tuning: -

## Advanced combination: -

# Development and Implementation: -

# Conclusion: -

# Erster Abschnitt des Hauptteils

## Hinweise zur Formatierung

In diesem Teil werden die verwendeten Formatvorlagen erläutert und auf allgemeine formale Anforderungen eingegangen. Die FV werden über den Pfeil am rechten unteren Rand im Reiter „Start – Formatvorlagen“ aufgerufen und erscheinen in einem extra Fenster.

Grundtext und zugehörige Formate:

Alle Passagen im Text sollten mit einer eigenen Formatvorlage (FV) formatiert sein.

Die FV sind so definiert, dass bei Eingabe einer Absatzmarke automatisch in die richtige neue Formatvorlage gewechselt wird.[[1]](#footnote-2)

* Der Text der Arbeit wird mit der Formatvorlage „Grundtext“ definiert.
* Anmerkungen werden in diesem Dokument beispielhaft als Aufzählung mithilfe der FV „Grundtext (Aufzählung)“ formatiert.
* Grob gegliedert unterteilt sich eine wissenschaftliche Arbeit in vier Bereiche, die durch eine angemessene Gliederung widergespiegelt werden müssen (Nummerierungen werden mit der FV „Grundtext (Nummerierung)“ formatiert):

1. Einleitung
2. Verwendete Grundlagen
3. Hauptsächliche Untersuchung/Entwicklung
4. Zusammenfassung und Ausblick

Überschriften:

* Zu jeder Überschrift aus den FV „Überschrift 1“ bis „Überschrift 3“ sollte mindestens eine halbe Seite Text folgen. Folgt einer Überschrift direkt eine tiefergestufte Überschrift (s. Kap. ‎1.2, so sollte der ersten Überschrift kein Text folgen; anstelle dessen sollte der Text der direkt folgenden tieferen Überschrift folgen.
* Folgen jedoch nur wenige Zeilen Text, oder ist der angegebene Text nicht von ähnlicher Bedeutung, wie die anstehende Überschrift, so kann die FV Zwischenüberschrift („ZwÜberschr1“) verwendet werden.
* Überschriften auf gleicher Gliederungsebene sollten stets die gleiche Bedeutung in Bezug auf das Thema haben.

Abkürzungen:

* Abkürzungen wie „z. B.“, „u. a.“, z. Zt.“ sind mit einem geschützten Leerzeichen ([Strg]+[Shift]+[Leertaste]) zu versehen. Erkennbar wird das geschützte Leerzeichen als „°„ nach Betätigen der Schaltfläche „¶“ in der Symbolleiste.
* Bei der ersten Verwendung einer Abkürzung ist es für den Leser hilfreich, die Abkürzung einmal ausgeschrieben im Text zu finden. Beispielsweise:

"Alle Passagen im Text sollten mit einer eigenen Formatvorlage (FV) formatiert sein.“

* Eine Überschrift oder ein Text kann mit der Tastenkombination [Shift]+[Alt]+[→] tiefer- bzw. durch [Shift]+[Alt]+[←] höher gestuft werden.

Grundzeichen (Grundz.) als Hervorhebung für einzelne Wörter

* Sollen einzelne Wörter im Text hervorgehoben werden, so ist eine kursive Hervorhebung dem Druck in fetter Schrift vorzuziehen.
* Mit „Grundz. (Sprache)“ wird ein fremdsprachlicher Text gekennzeichnet. Sollte eine Überprüfung des Abschnitts nicht erwünscht sein, so kann über die Vorlage „Grundz. (keine Sprache)“ die Rechtschreibung ausgestellt werden.
* Die Silbentrennung von Wörtern (im Besonderen von Fremdwörtern) kann durch die Ver­wen­dung geschützter Trennzeichen ([Strg]+[-]) manuell beeinflusst werden. Auf die korrekte Setzung der Trennstellen ist dabei eigenständig zu achten. Eine Silbentrennung im Englischen ist zu vermeiden.
* Bei der Verwendung von Bindestrichen zur Verkürzung der Schreibweise („Linien­art und –stärke“) ist ein geschützter Trennstrich ([Strg]+[Shift]+[-]) zu verwenden. So erfolgt auch bei einem anderen Zeilenumbruch eine korrekte Trennung und der Bindestrich steht nicht vereinzelt am rechten Rand.
* Gedankenstriche – wie in diesem Beispiel – werden über [Strg] + [-] auf dem Nummernblock gesetzt.
* Fehlerhaft formatierte Absätze (z. B. „Grundtext + Fett“ anstelle von „Grundz. (fett)“) können korrigiert werden, in dem der betreffende Absatz markiert wird, unter „Formatierung des markierten Textes“ der Anzeige „Formatvorlagen und Formatierung“ im Kontextmenü „Formatierung löschen“ gewählt wird und dem entsprechenden Absatz danach die richtige FV zugewiesen wird. Eine direkte Umformatierung des Absatzes wird von Word manchmal nicht angenommen.
* Bei der Umformatierung mehrerer gleicher Passagen kann in der Anzeige „Formatvorlagen und Formatierung“ bzw. „Start/Formatvorlagen“ im Kontextmenü (Rechtsklick) der betreffenden FV der Menüpunkt „Alle Instanzen von [Nummer] markieren“ gewählt werden.
* Mit der FV „Grundz. (Kapitälchen)“ werden die Namen von Autoren im Fließtext hervorgehoben; über „Grundz. (Courier)“ kann Programmcode o. ä. vom Fließtext unterschieden werden. Sollen jedoch ganze Abschnitte von Programmcode, z. B. SQL-Statements, in die Arbeit aufgenommen werden, so ist die FV „Grundtext (SQL)“ zu verwenden, die Schrittweite der Tabulatoren kann den eigenen Bedürfnissen angepasst werden.

SQL-Beispiel

SELECT Kunde.KdName, Bestellung.BestNr, Lieferant.LfNr

FROM Kd\_DATA AS Kunde, KDLFZUO AS Bestellung, LF0034 AS Lieferant

WHERE Bestellung.KdNr = Kunde.KdNr AND Bestellung.LfNr = Lieferant.LfNr AND Bestellung.Status = `OPEN´;

## Allgemeine Hinweise zur Benutzung von Word

### Makros

Durch die Verwendung von Makros können sich wiederholende Aufgaben automatisiert erledigt werden. Für das eigene Erstellen von Makros sei hier auf die Online-Hilfe verwiesen. Kurze Makros können per Maus aufgezeichnet und ggf. nachbearbeitet werden.

### Querverweise

Wird in der Arbeit auf andere Stellen verwiesen, so ist jeder Verweis immer durch einen Querverweis zu realisieren. Über „Einfügen/Links/Querverweis“ können einige vordefinierte Querverweise benutzt werden. Befindet sich der gewünschte Querverweis nicht in der Liste, so kann mit „Einfügen/Links/Textmarke“ an der ursprünglichen Stelle eine Textmarke definiert werden, auf deren Inhalt dann mit einem Textmarken-Querverweis zugegriffen werden kann.

* Sämtliche Verweise werden stets nur als „Kategorie und Nummer“ eingefügt, komplette Abschnitts‑ oder Abbildungsnamen werden nicht referenziert.
* Der Anhang ist über einen Querverweis stets als „Nummeriertes Element/Absatznummer (kein Kontext)“ zu referenzieren.
* Über die Tastenkombination [Shift]+[F9] werden die Feldfunktionen ein‑ und ausgeschaltet. Über [F9] wird ein Feld aktualisiert. Diese beiden Tastenkombinationen können durch vorheriges Markieren des gesamten Textes ([Strg]+a) auch auf das ganze Dokument angewendet werden.

# Abbildungen und Tabellen

## Verwendung von Abbildungen

Beim Einbinden mehrerer Abbildungen ist eine formale Hauptforderung die Einheitlichkeit der verwendeten Schriftgrößen. Zwei in gleicher oder vergleichbarer Notation erstellte Grafiken sollten auch dieselbe Schriftgröße verwenden. Soweit möglich sollten Abbildungen mit der Skalierung 100% in den Text eingebunden werden.

Grafiken sollten nach Möglichkeit selbst erstellt werden. Als Programm ist dabei i.d.R. Mircosoft Power Point zu nutzen. Auch aus anderen Werken übernommene Grafiken sollten nur dann in einer gescannten Version eingebunden werden, falls sie eine hohe Komplexität aufweisen und die Qualität des Scans ausreichend gut ist.[[2]](#footnote-3)

* Korrekte Abbildungen sind über die Registerkarte „Layout“ auf „Mit Text in Zeile“ formatiert, wie in Abb. ‎4.1 verdeutlicht.
* Über „Einfügen“, „Formen“ kann ein neuer Zeichenbereich eingefügt werden. Dadurch werden Abbildungen mit Formelementen zusammengefasst

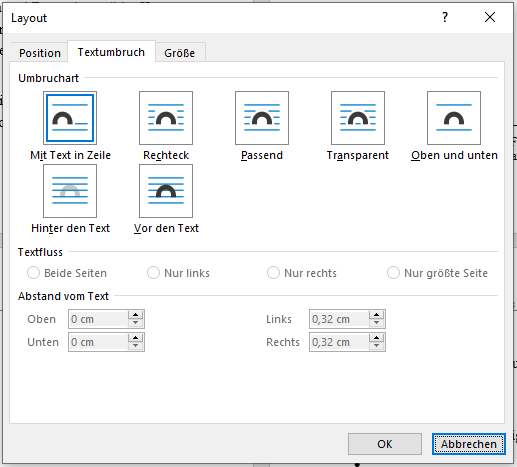


Abb. ‎4.1: Layoutoptionen in Word

Formatierung in Word

Eine Abbildung wird in Word mit der FV „Grafik“ formatiert. Anschließend folgt dann die Beschriftung mit der FV „Beschriftung“. Wenn die Grafik vollständig selbst erstellt ist und sie nicht einer Abbildung einer Quelle ähnelt, so erscheint keine Quelle. Die Angabe von „Quelle: eigene Abbildung“ ist falsch. Falls es sich um eine angepasste Quelle handelt schreibt man: (i.A.a. [X]).

Vorgehen

Zunächst wird die Abbildung eingefügt.

Die Grafik kann über „Beschriftung einfügen…“ im Kontextmenü der Grafik beschriftet werden. Dabei wird die Grafik in das Abbildungsverzeichnis eingebunden.

* Die Zeichenfolge „Abb. Kapitelnr.Abbildungsnr.“ muss fett (FV „Beschriftung“ + „Grundz. (fett)“) sein und der nachfolgende Text wieder dem Format der FV „Beschriftung“ (nicht fett) entsprechen. Getrennt werden die Teile durch einen Tabulator.
* Bei mehrzeiliger Abbildungsunterschrift ist auf die korrekte Ausrichtung zu achten (Tabulator, Einrückung).
* Grafiken sind stets mittig zu formatieren.

Beispiele:



Abb. ‎4.2: Funktionsdekompositionsdiagramm Marketing (Bec04)

In Tab. ‎4.1 sind die drei möglichen Kombinationen der Referenzierung von Abbildungen an einem Beispiel erläutert.

Tab. ‎4.1: Quellenangaben bei Abbildungen

|  |  |
| --- | --- |
| Beispiele für Quellenangaben | Bemerkung |
| [X] | Die Abbildung ist ohne Veränderungen übernommen worden. Sie ist mit dem Original identisch (und enthält sogar eventuelle Schreibfehler der Original-Quelle). |
| (i.A.a [X]) | An der Abbildung sind Veränderungen oder Ergänzungen vorgenommen worden. Diese können sowohl inhaltliche Korrekturen oder Ergänzungen, als auch formale Veränderungen (Sprache, Rechtschreibfehler etc.) beinhalten. |
| {keine Angaben} | Die Abbildung ist eine eigene Darstellung. Sie lehnt sich in keiner Weise an die grafischen Darstellungen anderer Autoren an. Die Angabe einer Quelle entfällt. |

* Bei der Erstellung von Abbildungen ist darauf zu achten, dass die erzeugten Grafiken selbstähnlich sind. D. h., Größe, Schriftart, Linienart und –stärke, sowie die Art der Pfeilspitzen sollten in allen Grafiken gleich gewählt werden.
* Auf Hervorhebungen wie Schatten und 3D-Effekte ist in der Regel zu verzichten.
* Die Lesbarkeit gefüllter Elemente nimmt mit zunehmendem Sättigungsgrad der Füllung ab. Daher sollte bei kleinen Schriftgrößen, wenn überhaupt, eine sehr helle Füllung verwendet werden.
* Grafiken müssen stets inhaltlich mit dem Text verknüpft sein; d. h., die Grafik muss einen inhaltlichen Bezug zur Arbeit aufweisen und muss in dieser auch erwähnt werden. Des Weiteren sollten Grafiken bezugsnah platziert werden.

## Verwendung von Tabellen

Tabellen in Word

Bei der Erstellung von Word-Tabellen wird die FV „Grundtext (Tabelle)“ auf alle Zellen angewendet. Spaltenüberschriften können zusätzlich mit der FV „Grundz. (fett)“ fettgedruckt formatiert werden. Die vorgegebene Schriftgröße 10pt dient als Empfehlung, je nach Umfang der Tabelle kann die Größe an die übliche Schriftgröße 12pt angepasst oder auf 8pt reduziert werden. Bei der Verwendung mehrerer Tabellen ist auf größtmögliche Selbstähnlichkeit zu achten.

Tabellen aus Excel

Auch bei der Einbindung von Excel-Tabellen ist auf größtmögliche Selbstähnlichkeit zu achten.

Die Tabellen aus Excel besitzen im Gegensatz zu anderen Abbildungen einen Rahmen; dieser wird bereits in Excel vergeben. Das von Excel standardmäßig erzeugte Gitternetz wird mit kopiert. Wird dies nicht gewünscht, so kann in Excel über „Ansicht/Einblenden/Ausblenden“ die Funktion „Gitternetzlinien“ ausgeschaltet werden.

Tab. ‎4.2 enthält eine Übersicht über Formatsempfehlungen für Abbildungen und Tabellen.

Tab. ‎4.2: Übersicht der Formatempfehlungen für Abbildungen und Tabellen

|  |  |  |
| --- | --- | --- |
|  | Abbildungen | Tabellen |
| Schriftart & ‑größe | Arial; 10pt für Überschriften, 8pt für Standardtext, 6pt für Anmerkungen | Arial, 10pt; Hervorhebungen über fette Schrift |
| Skalierung | i. d. R. keine, evtl. Hochskalierung kleiner Grafiken | --- |
| Linienstärke | 1pt; durchgezogene Linie | Standard (1pt) |
| Pfeilart & ‑größe | Pfeilart 4 (Dreieck); klein | --- |
| Rahmen | Kein Rahmen | Rahmen (gesamt); 1pt |

* Tabellen werden analog zu Abbildungen beschriftet, der einzige Unterschied besteht darin, dass die Beschriftung mit „Tab.“ statt „Abb.“ beginnt und oberhalb der Tabelle steht.
* Wenn möglich sollten Tabellen nicht durch Seitenumbrüche getrennt werden.
* Bei Tabellen über mehrere Seiten kann es sinnvoll sein, den Tabellenkopf auf jeder Seite mitzuführen. Dies wird erreicht über das Kontextmenü „Tabelleneigenschaften…“ Registerkarte „Zeile“, Funktion „Gleiche Kopfzeile auf jeder Seite wiederholen“ und Auswahl der gewünschten Zeile(n) über „Vorherige Zeile/Nächste Zeile“.

# Formeln

Formeln werden als Tabelle eingefügt und über die Maske „Verweise“ eingebunden:

|  |  |
| --- | --- |
|  | ( ‑ ) |

Im Text wird eine Formel durch einen Querverweis Formel ( ‎5‑1 ) eingebaut.

|  |  |
| --- | --- |
|  | ( ‑ ) |

Im Text wird eine Formel durch einen Querverweis Formel ( ‎5‑2 ) eingebaut.

Am besten wird diese Tabelle immer wieder kopiert und durch die neue Formel ersetzt.

Binden Sie auch Variablen und Einheiten als „Formeln“ (Einfügen – Formel) ein.

Variablen und Konstanten

Nutzen Sie bei der Verwendung von Formeln die Formatierungsgrundlage nach DIN 1338.

Variablen: – kursiv, kleine Buchstaben, nicht fett

Einheiten: – nicht kursiv, nicht fett (beachten Sie den Abstand zum Zahlenwert)

Vektoren: – kursiv, kleine Buchstaben, fett

Matrizen: – kursiv, große Buchstaben, fett

Funktionen: – nicht kursiv, nicht fett

# Zitieren mit Citavi

Bitte beachten Sie die separate Citavi Anleitung.

Als Zitierstil wird LNI verwendet.

Bei Werken, aus denen wörtlich oder sinngemäß zitiert wird, werden die Belegstellen in Kurzform angegeben – nach folgender Form:

* [SCH04]
  + SCH ist dabei der auf drei Buchstaben gekürzte Nachnamen des Erstautors (z.B. Schiller)
  + 04 ist das Jahr der Veröffentlichung (2004).

# Druck der Arbeit

Bachelor- und Masterarbeiten sind in 3-facher Ausfertigung auf einseitig bedrucktem Papier (DIN A4) in gebundener Form abzugeben. Auf besonders hochwertige Bindungen und Cover kann verzichtet werden.

# Zusammenfassung und Ausblick

Innerhalb der Zusammenfassung sollen die Ergebnisse der Arbeit in Hinblick auf das Ziel zu Beginn zusammengefasst werden. Weiterhin sollen weitergehende Forschungen aufgezeigt werden.

# Literaturverzeichnis

Über Citavi oder „Verweise“ – „Literaturverzeichnis“ einfügen. Die gängige Vorlage ist LNI.

Fußnoten als Quellen Angaben sind unüblich!

# Anhang

Detailergebnisse die nicht unmittelbar im Hauptteil vorhanden sein müssen (z.B. weitere Messergebnisse, Simulationsergebnisse, technische Zeichungen, …)

# Eidesstattliche Versicherung

|  |  |  |  |
| --- | --- | --- | --- |
| Name: |  | Vorname: |  |
| Matrikel-Nr.: |  | Studiengang: |  |

Hiermit versichere ich, .< Vorname, Name >., an Eides statt, dass ich die vorliegende < Bachelor- oder Masterarbeit > mit dem Titel < Titel der Abschlussarbeit > selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken ent­nommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der eidesstattlichen Versicherung und prüfungsrechtlichen Folgen sowie die strafrechtlichen Folgen (siehe unten) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.

## Auszug aus dem Strafgesetzbuch (StGB)

***§ 156 StGB:*** Falsche Versicherung an Eides Statt

Wer von einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Ort, Datum Unterschrift

Publication bibliography

Anyscale (2023): What is Hyperparameter Tuning? | Anyscale. Available online at https://www.anyscale.com/blog/what-is-hyperparameter-tuning, updated on 8/12/2023, checked on 8/12/2023.

Bertoni, Alessandro (2018): Data-Driven Design in the Product Innovation Process: Role and Research Challenges, checked on 2018.

din-en (2023): A brief introduction to standards. Available online at https://www.din.de/en/about-standards/a-brief-introduction-to-standards, updated on 9/16/2023, checked on 9/16/2023.

Dunjko, Vedran; Briegel, Hans J. (2017): Machine learning \& artificial intelligence in the quantum domain. Available online at http://arxiv.org/pdf/1709.02779v1.

Giering, Oliver (2022): Künstliche Intelligenz und Arbeit: Betrachtungen zwischen Prognose und betrieblicher Realität. In *Z. Arb. Wiss.* 76 (1), pp. 50–64. DOI: 10.1007/s41449-021-00289-0.

Ioannis, ZACHARIADIS (2019): Standards and the digitalisation of EU industry, checked on 2019.

ISO (2023): Enabling an AI-ready culture. Available online at https://www.iso.org/news/ref2763.html, updated on 9/16/2023, checked on 9/16/2023.

Kili Technology (2022): Your Guide To AI Document Analysis, 2/18/2022. Available online at https://kili-technology.com/data-labeling/your-guide-to-ai-document-analysis, checked on 9/16/2023.

LOIBL, André; MANOHARAN, Thivakar; Nagarajah, Arun (2020): Procedure for the transfer of standards into machine-actionability. In *JAMDSM* 14 (2), JAMDSM0022-JAMDSM0022. DOI: 10.1299/jamdsm.2020jamdsm0022.

Luttmer, Janosch; Ehring, Dominik; Pluhnau, Robin; Nagarajah, Arun (2021): REPRESENTATION AND APPLICATION OF DIGITAL STANDARDS USING KNOWLEDGE GRAPHS. In *Proc. Des. Soc.* 1, pp. 2551–2560. DOI: 10.1017/pds.2021.516.

Luttmer, Janosch; Prihodko, Vitalijs; Ehring, Dominik; Nagarajah, Arun (2023): Requirements extraction from engineering standards – systematic evaluation of extraction techniques. In *Procedia CIRP* 119, pp. 794–799. DOI: 10.1016/j.procir.2023.03.125.

Nabi, Javaid (2019): Hyper-parameter Tuning Techniques in Deep Learning - Towards Data Science. In *Towards Data Science*, 3/16/2019. Available online at https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8, checked on 9/10/2023.

Rigger, Eugen; Vosgien, Thomas (2018): DESIGN AUTOMATION STATE OF PRACTICE - POTENTIAL AND OPPORTUNITIES. In : Proceedings of the DESIGN 2018 15th International Design Conference. 15th International Design Conference, May, 21-24, 2018: Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Croatia; The Design Society, Glasgow, UK (Design Conference Proceedings), pp. 441–452.

Rosebrock, Adrian (2021): Grid search hyperparameter tuning with scikit-learn (GridSearchCV ) - PyImageSearch. Available online at https://pyimagesearch.com/2021/05/24/grid-search-hyperparameter-tuning-with-scikit-learn-gridsearchcv/, updated on 5/7/2021, checked on 9/6/2023.

scikit-learn (2023): 3.1. Cross-validation: evaluating estimator performance. Available online at https://scikit-learn.org/stable/modules/cross\_validation.html, updated on 9/11/2023, checked on 9/11/2023.

Vincent, Amala Mary; Jidesh, P. (2023): An improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms. In *Scientific reports* 13 (1), p. 4737. DOI: 10.1038/s41598-023-32027-3.

Vitalijs Prihodko (2022): Eine Dokumentvorlage für Abschlussarbeiten und andere wissenschaftliche Arbeiten, insbesondere Bachelorarbeiten, Masterarbeiten, Abschlussarbeiten und Studienarbeiten.

www.javatpoint.com (2023): Genetic Algorithm in Machine Learning - Javatpoint. Available online at https://www.javatpoint.com/genetic-algorithm-in-machine-learning, updated on 9/6/2023, checked on 9/6/2023.

1. Diese Einstellung kann geändert werden, in dem unter „Formatierung des markierten Textes“ im Fenster „Formatvorlagen (Word 2007) die FV mit rechts angeklickt wird, im Kontextmenü „Ändern“ gewählt wird und im Punkt „Formatvorlage für Folgeabsatz“ die gewünschte FV eingestellt wird. [↑](#footnote-ref-2)
2. Ausreichend heißt in diesem Fall, dass der Grafik im ausgedruckten Format nicht unmittelbar angesehen werden kann, dass sie aus einem anderen Werk herauskopiert wurde. [↑](#footnote-ref-3)