TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Klassifikation von Monitoringdatenreihen mittels Machine-Learning

Ein Feature-basierter Ansatz

ausgeführt zum Zwecke des akademischen Grades eines „Diplom-Ingenieurs“ unter der Leitung von

Ao. Univ. Prof. Dipl.-Ing. Dr.techn Burkhard Kittl
und
Dipl.-Ing. Benjamin Mörzinger, BSc

E311
Institut für Fertigungstechnik und Hochleistungslasertechnik

eingereicht an der Technischen Universität Wien
Fakultät für Maschinenwesen und Betriebswissenschaften

von
Thomas Zeitelhofer, BSc
1170 Wien

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Eidesstaatliche Erklärung

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin. Ich erkläre weiters an Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Arbeiten selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachter) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum, Ort

Unterschrift

Danksagung

Mein besonderer Dank gilt meinen Eltern, die mir mein Studium ermöglicht und mich während der ganzen Zeit sowohl mental als auch finanziell unterstützt haben.

Herzlich bedanken möchte ich mich bei meiner Freundin Julia, die mich immer wieder ermutigte und mit ihren zahlreichen Korrekturarbeiten einen wesentlichen Teil zu dieser Diplomarbeit beigetragen hat.

Weiterer Dank gilt meinem Betreuer Dipl.-Ing. Benjamin Mörzinger, der diese spannende Diplomarbeit ermöglicht hat und mich bestens während der Erstellung dieser Arbeit betreut hat.

Zu guter Letzt möchte ich mich bei meinen Freunden für ihre stets hilfreiche Unterstützung und für eine sehr schöne Studienzeit bedanken.

Kurzfassung

Durch die Digitalisierung befindet sich die gesamte fertigende Industrie im Wandel. In der Produktion wird immer mehr Sensorik verwendet, um Optimierungspotentiale in Fertigungsprozessen auffinden zu können. Die Menge an Daten die hierbei anfällt ist, oftmals unüberschaubar und wird daher kaum bis gar nicht genutzt. Eine wichtige Gruppe von Produktionsdaten stellt die Gruppe der Zeitreihendaten dar. In dieser Arbeit wird eine Software Applikation vorgestellt, mit der es möglich ist, für die UserInnen relevante Zeitreihenabschnitte zu markieren. Die UserInnen bekommen keine Vorgaben, nach welchen Kriterien die Bewertung zu erfolgen hat, sondern sollen ihre Erfahrung auf dem Gebiet nutzen, um die Relevanz der Abschnitte festzulegen.

Die generierten Informationen dienen als Trainingsgrundlage für einen Machine Learning Algorithmus. Dieser Ansatz wird gewählt, da künstliche Intelligenz schon in anderen Bereichen, wie beispielsweise der Suche nach Forschungsarbeiten (`http://www.arxiv-sanity.com/`), erfolgreich zum Einsatz gekommen ist. Durch das Feedback des/der User-s/in soll der Machine Learning Algorithmus lernen, welche Datenreihen diese als relevant einstuft und welche nicht. Dadurch soll der Algorithmus in der Lage sein, neue Datenreihen richtig zu kategorisieren.

Ein wichtiger Bereich des Machine Learnings ist das Feature Engineering. Es werden drei verschiedene Featuresets zum Trainieren des Machine Learning Algorithmus eingesetzt: inhärente, generierte und kombinierte Features.

Ziel dieser Arbeit ist es, das Verhalten eines Machine Learning Algorithmus bei Verwendung verschiedener Featuresets zu zeigen. Zu Beginn wird überblicksartig die Thematik Machine Learning und Feature Engineering erläutert. In einem weiteren Schritt wird die Methodik und Implementierung der Applikation behandelt, wobei auf die Auswahl der Features detaillierter eingegangen wird. Abschließend wird eine Lernkurve auf Basis der Genauigkeit erstellt. Es wird gezeigt, dass die Genauigkeit durch das Training mit dem kombinierten und inhärenten Featureset besser ist, als durch das Training mit dem generierten Featurset.
Ein weiteres Ergebnis dieser Arbeit ist, dass Machine Learning für Aufgaben der Relevanzerkennung geeignet ist. Dadurch ist es möglich, Leuten Empfehlungen für relevante Datenreihen geben zu können. Durch die Variation des Machine Learning Algorithmus und der Features eröffnen sich weitere Forschungsthemen auf diesem Gebiet.

Abstract

Due to the digitization the whole manufacturing industry is changing. To find potential improvements, a rising amount of sensors are used in production. The high quantity of data which is generated is often incomprehensible and therefore hardly used. An important area of production data is the group of timeseries data. In this master thesis a software application is created, which allows the user to mark relevant sections of timeseries data. There are no specifications for the user, to mark the sections. The user shall use his experience to decide whether the time series segment is relevnat or not.

The generated information is used as the base for training of a machine learning algorithm. This approach is chosen, because artifical intelligence has successfully been used in other areas, like in searching for research papers in the internet (`http://www.arxiv-sanity.com/`). The aim of the training is that the machine learning algorithm is able to decide whether a new segment is potentially relevant or not.

An important part of machine learning is feature engineering. Three sets of features are used for the training of the algorithm: inherent, generated and combined featuresets.

The goal of this work is to show the behavior of the machine learning algorithm due to the different sets of features. At the beginning of the work the topics machine learning and feature engineering are covered. Furthermore the methods and the implementation of the application are explained. The work focuses on the different sets of features and on feature engineering itself. Finally a learning curve based on the calculated accuracy is created. It is demonstrated that the accuracy due to the training with the combined and inherent featureset is higher than due to the training with the generetad featureset. An additional result of this work is, that machine learning is a suitable tool for tasks of relevance recognition. Due to this, it is possible to give people recommondations for relevant sequences of data. Through the variation of the machine learning algorithm and the features there are opportunities for further research topics.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Problemstellung & Motivation 2

1.2 Zielsetzung 3

2 Grundlagen **5**

2.1 Allgemeines 5

2.2 Entwicklung & Stand der Technik 6

2.3 Machine Learning 10

2.3.1 Aufgabe T 10

2.3.2 Erfahrung E 13

2.3.3 Performance P 13

2.4 Feature Engineering 16

2.5 Ermittlung relevanter Zeitreihendaten 19

2.5.1 Relevance User Feedback in Zeitreihendaten 19

2.5.2 Ansätze zur Identifizierung relevanter Zeitreihen 20

3 Aufbau und Beschreibung der Applikation **23**

3.1 Methodik und Struktur 24

3.2 Auswahl der Features 27

3.2.1 Inhärente Features (grüne Features) 28

3.2.2 Generierte Features (rote Features) 29

3.2.3 Kombinierte Features (blaue Features) 32
3.3 Auswahl der Machine Learning Methodik 33
3.4 Implementierung des Machine Learning Segments 35

4 Auswertung der Daten 42

5 Fazit und Ausblick 45

Literaturverzeichnis 50
Abbildungsverzeichnis 52
Tabellenverzeichnis 53

A Anhang I

Nomenklatur

Symbol	**Eigenschaften**
TP	„True Positive“ - Wert
TN	„True Negative“ - Wert
FP	„False Positive“ - Wert
FN	„False Negative“ - Wert
ϵ	Genauigkeit
ρ	Präzision
π	Sensitivität/Trefferquote
F_1	Fehlerraten
μ	Mittelwert
σ	Standardabweichung
ν	Schiefe
ω	Wölbung

1 Einleitung

In den nächsten Jahren wird die Wertschöpfung durch Wissen und Informationen aus Daten eine bedeutsame Rolle in Industrieunternehmen spielen. Durch die Analyse von Daten sollen Kosten-, Effizienz- und Zeitvorteile für die Unternehmen erzielt werden. [1] Heutzutage gibt es Ansätze, die durch Datenanalysen Ertragssteigerungen in der Produktion erzielen. [2], [3], [4]
Produktionsanlagen stellen hierbei bereits gute Datenquellen dar. Durch moderne Sensorik können diese zusätzlich in Netzwerke integriert werden. Dadurch ist unter anderem die Kontrolle der Produktionsleistung und Produktionsqualität möglich. Zukünftig soll es eine flächendeckende Verbindung zwischen der digitalen und physischen Welt geben und dadurch die Kommunikation von Systemen untereinander möglich sein. [5]
Die Verknüpfung von physischen Dingen mit einem Netzwerk, nennt man „Internet of Things“. Ziel ist es, die gesammelten Daten zu verknüpfen und im Netzwerk zur Verfügung zu stellen. [5] Nicht nur neue Maschinen haben einen Netzwerkzugriff, sondern auch ältere Geräte können mit der passenden Sensorik nachgerüstet und mit dem Internet verbunden werden. Diese Sensoren können potentiell eine sehr große Menge an unterschiedlichsten Daten erzeugen, die oft zeitnah verarbeitet werden müssen, um daraus nutzbare Informationen generieren zu können. [5]

Aufgrund der sehr großen Datenmenge ist diese von Menschen nur mehr sehr schwer zeitnah zu analysieren und auszuwerten. Diese Arbeit setzt sich mit der Klassifizierung maschineller Produktionsdaten auseinander. Dabei wird eine Methodik vorgestellt, mit der UserInnen Abschnitte von Produktionsdatenreihen nach deren Relevanz kennzeich-

nen können. In nächsten Schritt wird diese Information automatisch weiterverarbeitet, um Muster in den Produktionsdaten auffinden zu können.
Hierfür wird der Ansatz des maschinellen Lernens gewählt, da es ein gutes Werkzeug darstellt, das auf Basis der UserInnendaten Vorhersagen für neue Datensätze machen kann. Dieser Ansatz wird gewählt, da dieser in anderen Bereichen schon erfolgreich zum Einsatz gekommen ist. Ein Beispiel ist die Suche nach wissenschaftlichen Arbeiten im Internet (`http://www.arxiv-sanity.com/`). Dabei werden anhand vergangener Suchbegriffe und der Ähnlichkeit zu anderen Arbeiten weitere passende Arbeiten vorgeschlagen. Durch das Training mit den Datensätzen aus dem UserInnen Feeback sollen die Algorithmen des Machine Learnings verstehen, welche Daten für die UserInnen potentiell relevant sind.

1.1 Problemstellung & Motivation

Durch den Einsatz moderner Sensorik in der Produktion steigt die Menge an gespeicherten Daten ständig. Eine wichtige Kategorie der gespeicherten Daten sind Zeitreihendaten. „Wenn ein Merkmal X zu mehreren aufeinanderfolgenden Zeiten t_1, t_2,..., t_n (Zeitpunkte oder Perioden) beobachtet wurde, bezeichnet man die Daten als Zeitreihe und schreibt die Beobachtungen $X(t_1)$, $X(t_2)$,..., $X(t_n)$ [...] “ [6, S.14] Unter diese fallen beispielsweise Temperatur-, Kraft-, Geschwindigkeits- und Leistungsdatenreihen.
Die Problematik besteht darin, dass die gespeicherte Datenmenge kaum bis gar nicht genutzt wird, da der Mensch nicht mehr in der Lage ist, aus der großen Menge an Daten die relevanten Abschnitte zu finden. Um das Wissen aus den Daten zeitsparend extrahieren zu können ist es nötig, diese maschinell unter Zuhilfenahme von intelligenten Algorithmen zu verarbeiten.
Ein geeignetes Werkzeug ist der Einsatz von künstlicher Intelligenz bzw. maschinellem Lernen. Verschiedene Ansätze haben gezeigt, dass der Einsatz von Machine Learning in der Klassifikation von Zeitreihendaten unter Zuhilfenahme von UserInnen-Feedback erfolgreich sein kann. [7], [8], [9] Dabei besteht das Feedback der UserInnen darin, die relevanten Zeitreihendatenabschnitte als „relevant“ oder „irrelevant“ zu kennzeichnen. Diese Information dient als Metainformation, die den Algorithmen der künstlichen Intelligenz

übergeben werden, damit dieser weiß, in welche zwei Gruppen die Zeitreihenabschnitte zu kategorisieren sind. Zusätzlich zu der Relevanz-Information werden Informationen benötigt, die die relevanten Zeitreihenabschnitte identifizieren (siehe Abbildung 1.1).

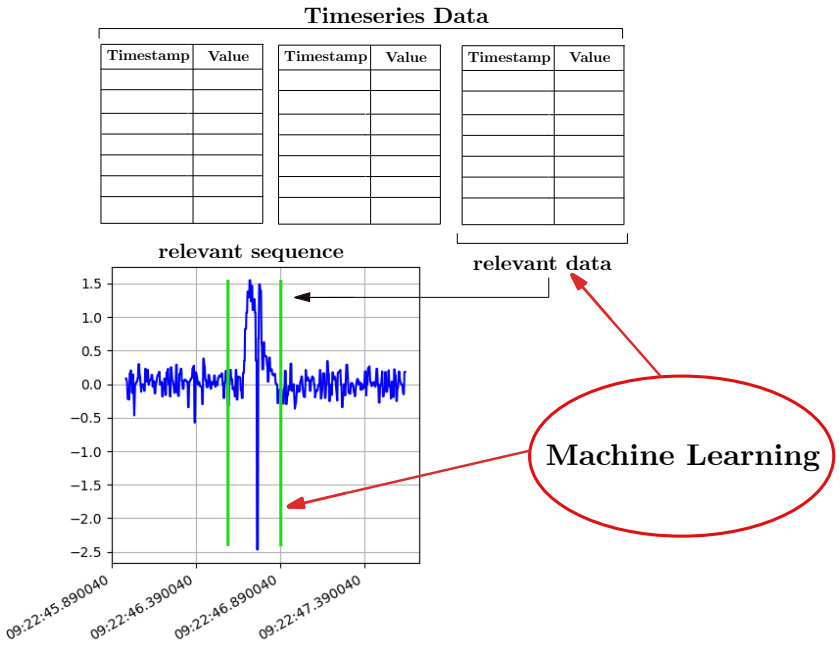

Abb. 1.1: Grafische Darstellung der Problemstellung

Diese Metainformationen werden im maschinellen Lernen als Features bezeichnet und stellen einen wichtigen Bereich dar. Die richtige Auswahl der Features ist maßgeblich für den Erfolg einer Aufgabenstellung für maschinelles Lernen. [10]

1.2 Zielsetzung

Im Zuge dieser Arbeit soll folgende Forschungsfrage beantwortet werden:

„Wie beeinflussen unterschiedliche Feature-Sets den Lernprozess eines Machine Learning Algorithmus bei der Klassifizierung von Monitoringdaten? “

Zur Beantwortung dieser Frage soll in dieser Arbeit ein Softwaretool entwickelt werden, das den UserInnen Datenreihen präsentiert. Folglich soll entschieden werden, ob die Datenreihen „relevant“ oder „irrelevant“ sind. Im Hintergrund werden Datensätze erzeugt,

die dem Algorithmus zum Trainieren übergeben werden. Abschließend wird das Verhalten des Algorithmus nach dem Training mit verschiedenen Datensätzen analysiert, und anhand von Kennwerten bewertet, wie gut der Lernerfolg durch die verschiedenen Features ist.

2 Grundlagen

Neue Methoden und Analyseverfahren, um Zeitreihendaten effizienter verarbeiten und darstellen zu können, sind regelmäßig Inhalt von Forschungsfragen. [7], [11], [9], [8], [12], [13] In diesem Kapitel werden die Grundzüge der Thematik Machine Learning und der Stand der Technik erläutert. Zusätzlich werden verschiedene Methoden und Techniken aus der Forschung beschrieben um relevante Zeitreihendaten aufzufinden.

2.1 Allgemeines

Mithilfe von Computern ist es heutzutage sehr einfach, Daten zu speichern. Die Entscheidung über die Weiterverarbeitung der Daten wird oftmals aufgeschoben, da Speichermedien und online Speicherorte nicht teuer sind. Das Problem ist, dass potenziell nützliche Informationen in der gespeicherten Datenmenge verborgen liegen und kaum Vorteile daraus gezogen werden. Data Mining versucht versteckte Muster zu finden. Die Mustersuche ist an sich nichts Neues, aber dadurch, dass diese Suche in Datenbanken automatisch erfolgt und durch die steigende Datenmenge die Möglichkeit steigt, Muster in dieser zu finden, wird Data Mining immer mehr Bedeutung zugesprochen. [14] „Intelligently analyzed data is a valuable resource. It can lead to new insights, better decision making, and, in commercial settings, competitive advantages.“ [14, S.5]

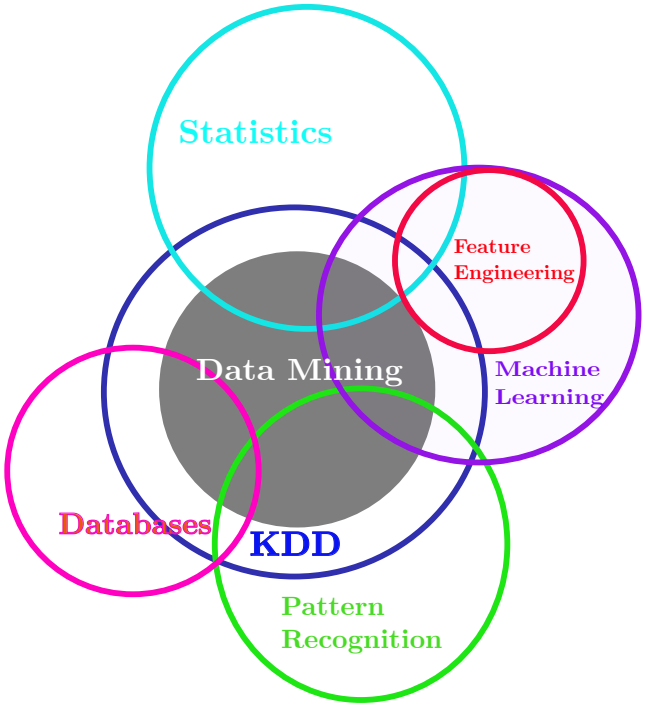

Abb. 2.1: Venn-Diagramm Data Mining, vgl. [15]

Der Begriff Data Mining beschreibt eine Methode zur automatischen Generierung von Wissen aus großen Mengen von Daten. Diese neu gewonnenen Erkenntnisse sollen potenziell nützlich und verständlich sein. Data Mining ist ein multidisziplinärer Bereich, den man auch in anderen Bereichen wie Machine Learning und Statistik wiederfindet (siehe Abbildung 2.1). Diese zwei Bereiche liefern zudem wichtige Werkzeuge, die im Bereich Data Mining zum Einsatz kommen. [5] Zunächst wird die Entwicklung des Machine Learning und der Stand der Technik erläutert. Im Weiteren wird auf den Bereich Machine Learnings, insbesondere auf den Bereich Feature Engineering, eingegangen.

2.2 Entwicklung & Stand der Technik

Das Verlangen nach denkenden Maschinen gibt es schon seit einiger Zeit. Schon damals als die ersten Computer entworfen wurden, stellte man sich die Frage, ob diese Maschinen womöglich intelligent werden können. Heutzutage ist künstliche Intelligenz („Artificial Intelligence“ - AI) ein aufstrebendes Gebiet mit vielen praktischen Anwendungen und

Forschungsgebieten. [16] Abbildung 2.2 gibt einen Überblick über die zeitliche Entwicklung des Machine Learnings.

„In the early days of artificial intelligence, the field rapidly tackled and solved problems that are intellectually diffcult for human beings but relatively straight- forward for computers—problems that can be described by a list of formal, mathematical rules. “ [16, S.1] Die Aufgaben, die mittels künstlicher Intelligenz gelöst werden sollten, sind für Menschen leicht auszuführen. Formal sind sie jedoch schwer zu beschreiben, wie beispielsweise das Erkennen von Gesichtern in Bildern. [16]

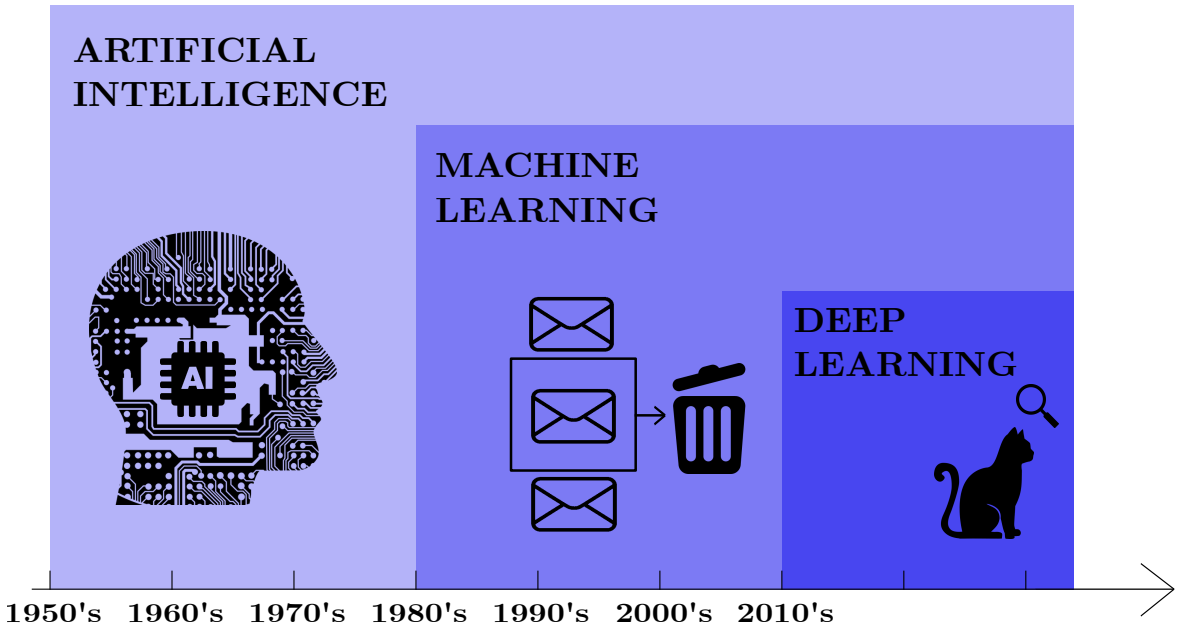

Abb. 2.2: Entwicklung des Machine-Learnings, vgl. [17]

Diverse Projekte, die sich mit künstlicher Intelligenz beschäftigt haben, haben versucht, das bestehende Wissen in formalen Sprachen abzuspeichern. Computer können diese formalen Statements nutzen um daraus automatisch, mithilfe von logischen Regeln, Schlussfolgerungen zu ziehen. Dies wird auch Wissensbasis-Ansatz genannt. Die Projekte, die sich mit diesem Ansatz auseinandersetzen, waren nicht erfolgreich. [16]
Das Problem, das sich herauskristallisiert hat, war, dass Systeme mit künstlicher Intelligenz die Fähigkeit brauchen ihr eigenes Wissen aus den Mustern der Daten zu generieren. Diese Fähigkeit führt zum Begriff des Machine Learnings. Die Einführung des Machine

Learnings ermöglichte es, Probleme mithilfe von Daten aus der realen Welt zu lösen. Ein einfacher Algorithmus namens „naive Bayes“ kann e-Mails in ordnungsgemäß oder als „spam“ einordnen (siehe Abbildung 2.2) Die Performance der Algorithmen hängt maßgeblich von der Darstellung der Daten ab, die zur Verfügung stehen. Die Informationen, die dann aus den Daten extrahiert werden, werden „Features“ genannt. [16]
Viele Aufgaben, bei der künstliche Intelligenz zum Einsatz kommt, können mithilfe der richtigen Features, die einem simplen Algorithmus übergeben werden, gelöst werden. Die richtige Wahl der Features stellt in vielen Aufgabenstellungen eine Herausforderung dar. Beispiele für Aufgaben inkl. nützlicher Features sind:

- Aufgabe: Klassifikation eines Sprechers nach Geschlecht und Alter anhand von Tonaufnahmen. Feature: Größe des Vokaltraktes [16]
- Aufgabe: Klassifikation von Angestellten. Feature: Alter [18]
- Aufgabe: Klassifikation der Blume „Iris“. Feature: Größe des Kronblattes [14]

Sollten die Tonaufnahmen im Beispiel der Klassifikation eines Sprechers durch einen Akzent des Sprechers verzerrt werden und möchte man diese Information extrahieren, benötigt es für die Generierung der Features eine anspruchsvolle Analyse bzw. ein fast menschenähnliches Verständnis der Daten. Dieses Problem kann mittels Deep Learning gelöst werden. Deep Learning ist auch unter der Bezeichnung „künstliche neuronale Netzwerke“ (KNN) bekannt. [18]
Durch Deep Learning können Computer aus simplen Featurekonzepten komplexere Konzepte bilden. Die Algorithmen arbeiten auf Basis verschiedener Darstellungen der Daten, die in mehreren Stufen gelernt werden. Diese geschichtete Struktur zeichnet einen Deep Learning Algorithmus aus. Die benötigten Features werden automatisch generiert. [16]
Ein Beispiel eines Anwendungsfalles für Deep Learning ist die Kategorisierung von Bildern. Es ist schwierig die Information der rohen Input Daten in der Form von Pixeln zu verstehen. Die Funktionen, die zur direkten Bearbeitung von Pixeln zur Objektidentifizierung benötigt werden, sind kompliziert. Der Deep Learning Algorithmus erstellt eine Abfolge von einfacheren Funktionen, die in einer Struktur mit mehreren Ebenen abgebildet werden. Der Input erfolgt in der ersten Ebene. Da die Variablen in dieser Ebene von dem/der User/in betrachtet werden können, wird diese als „visible layer“ (einsehbare Ebene) bezeichnet. Die nachfolgenden „hidden layer“ (versteckte Ebene) generieren in

jeder Stufe komplexere Features aus dem Bild. Die Werte dieser Ebenen sind nicht in den Daten abgebildet. Das Modell wählt ein passendes Konzept um Beziehungen in den untersuchten Daten herzustellen. [16]
Abbildung 2.3 illustriert die einzelnen Ebenen des Deep Learning Modells.

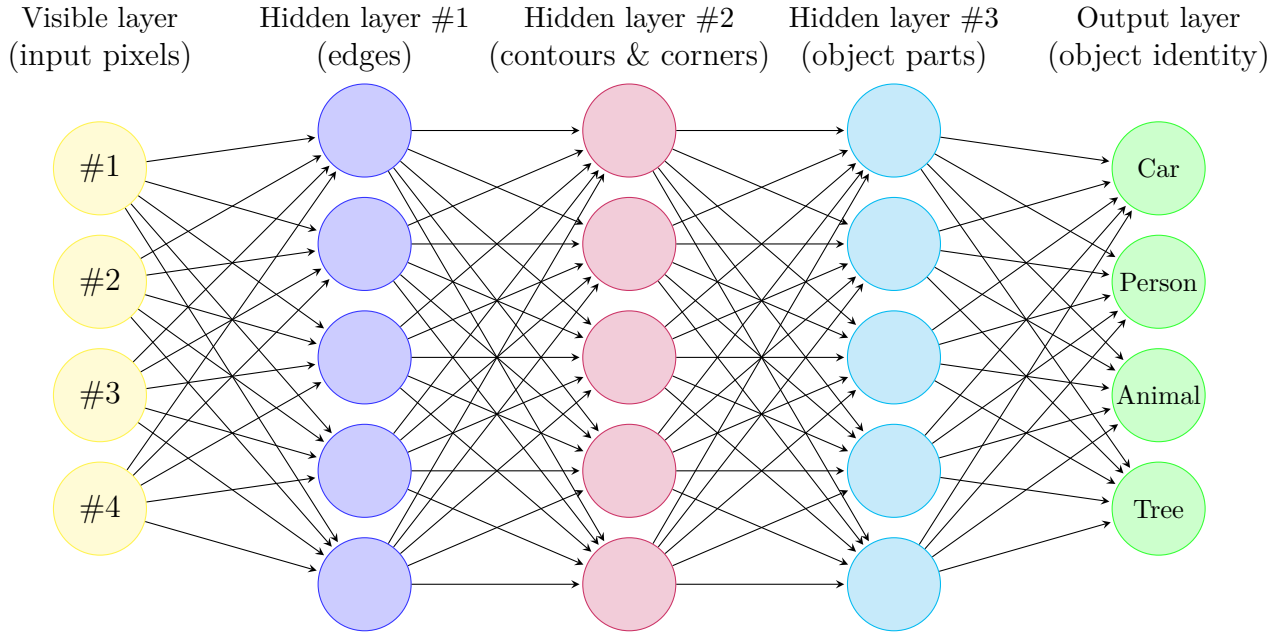

Abb. 2.3: Schematische Darstellung eines Deep Learning Modells (Kategorisierung eines Bildes), vgl. [16]

Deep Learning ist in der heutigen Zeit in vielen Bereichen vertreten. [14] Forschungsgebiete, in denen Deep Learning vertreten ist, sind vor allem Sprach- und Bilderkennung. [18]

2.3 Machine Learning

Eine formelle Definition von Machine Learning wurde im Jahr 1997 von Tom Mitchell aufgestellt: [18]

„A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.“ [18, S.9]

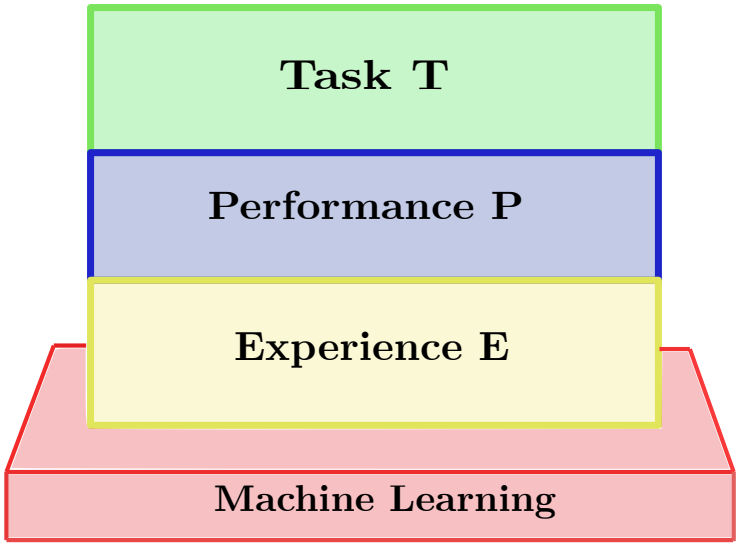

Abb. 2.4: Darstellung des ML-Modells

Wenn man diese Definition betrachtet, kann man vereinfacht sagen, dass die Algorithmen des maschinellen Lernens ihre Performance P verbessern, indem eine gewisse Aufgabe T mit der Erfahrung E eine Zeit lang durchgeführt wird. Die 3 Komponenten P, T und E sind die Hauptkomponenten des Lernprozesses. Abbildung 2.4 ist eine grafische Darstellung des Machine Learning Modells.

2.3.1 Aufgabe T

Die Aufgabe T des Machine Learnings Modells kann aus zwei Sichtweisen betrachtet werden. Einerseits kann man die Aufgabe aus der Sichtweise der eigentlichen Problem-

stellung betrachten. Dabei handelt es sich um das Problem aus der realen Welt, das durch das maschinelle Lernen gelöst werden soll. „Machine Learning based tasks are difficult to solve by conventional and traditional programming approaches.“ [18, S.10] Um eine geeignete Lösung mittels Machine Learning zu erreichen, muss die Problemstellung so genau wie möglich formuliert werden, um folglich daraus eine spezifische Machine Learning Aufgabe abzuleiten. [18]

Andererseits kann die Aufgabe T basierend auf dem Systemworkflow, der vorgibt, welche Schritte bei der Bearbeitung der Datensätze durchgeführt werden müssen, definiert werden. Diese Datensätze besitzen üblicherweise Eigenschaften, die für das Training der Algorithmen des maschinellen Lernens erforderlich sind. [18]
Je nach Typ der Lernmethodik können die Machine Learning Algorithmen in zwei Gruppen eingeteilt werden: Supervised Learning und Unsupervised Learning.

Supervised Learning

Die Methoden, die nach dem Supervised Learning Prinzip arbeiten, setzen die Input Variablen (unabhängige Variable) mit den Zielvariablen (abhängige Variable) in Beziehung. Diese Beziehung wird als Struktur abgebildet, die als Modell bezeichnet wird. Diese Modelle beschreiben Zusammenhänge, die zur Vorhersage von Werten der Zielvariablen bei gegebenen Eingangsvariablen dienen. Diese Art von Methodik findet im Finanzbereich, im Marketing sowie in der fertigenden Industrie Anwendung. [19]
Man kann zwischen zwei Hauptgruppen unterscheiden: Klassifizierungsmodelle („classifiers“) und Regressionsmodelle. [19]

- **Regression**
  Bei der Regression handelt es um eine Vorhersage eines numerischen Wertes. Ein Beispiel für eine reale Problemstellung ist die Vorhersage des Preises eines Hauses anhand der Angabe der Fläche, der Anzahl der Stockwerke und Zimmer als Input Daten. [18]

- **Klassifikation**
  Bei dieser Art von Machine Learning Aufgabe wird ein Datensatz übergeben, dem anschließend durch den Machine Learning Algorithmus eine spezifische Gruppe

oder Kategorie zugewiesen wird. Ein einfaches Beispiel ist die Klassifizierung von Bildern in Katzen- oder Hundebilder. [18]

„Along with regression and probability estimation, classification is one of the most studied models, possibly one with the greatest practical relevance. “ [19, S.133]

Unsupervised Learning

Die wichtigste Gruppe des Unsupervised Learnings ist das Clustering. Supervised Modelle, wie die Klassifizierung, sind voraussagend, wohingegen Unsupervised Modelle beschreibend sind. [19] Die Gruppen oder Cluster werden aus den Input Datensätzen gebildet, indem der trainierte Machine Learning Algorithmus Ähnlichkeiten, Beziehungen oder Muster in den Input Daten findet. Beispiele sind Gruppierungen von ähnlichen Einheiten, Produkten und Ereignissen. Die Unsupervised Learning Modelle besitzen gegenüber den Supervised Modellen den Vorteil, dass sie keine Trainingsdaten benötigen. Oftmals sind keine Trainingsdaten verfügbar und man möchte trotzdem Einsicht oder nützliche Muster in den Daten erkennen. Ein Objekt stellt eine Einheit der verfügbaren Datenmenge dar. [18] Die verschiedenen Clustering Modelle können wie folgt kategorisiert werden:

- **Hierachische Methoden**
  Diese Methode bildet die Cluster nach einem top-down oder bottom-up Ansatz. Die Kriterien zur Clusterbildung ergeben sich aus Merkmalen, die durch Affinitätsfunktionen bestimmt werden. [19] Dabei kann in zwei weitere Unterkategorien unterschieden werden:
  - **Agglomeratives hierarchisches Clustering**: Zu Beginn bildet jedes Objekt für sich einen eigenen Cluster. Anschließend werden die Cluster stufenweise solange zusammengefügt, bis die gewünschte Struktur erreicht ist. [19]
  - **Teilendes hierarchisches Clustering**: Ausgehend von einem Cluster, werden weitere Sub-Cluster gebildet, bis die angestrebte Struktur erreicht ist. [19]
- **Aufteilungsmethoden**
  Diese Methodik verschiebt von einer Anfangsaufteilung ausgehend jedes Objekt

von einem Cluster in einen anderen. Typischerweise benötigen diese Methoden die Anzahl der Cluster, die der/die User/in vorgibt. Um in dieser Methodik eine optimale Lösung zu erhalten, müssen alle möglichen Clusterkombinationen aufgelistet werden. Da dies nicht ausführbar ist, kommen iterative Verfahren zum Einsatz, die die Objekte zwischen den k Gruppen verschieben. [19]

- **Modell-basierte Methodik**
  Mithilfe von mathematischen Modellen werden die Objekte in Cluster eingeteilt. Anders als bei einem konventionellen Clustering, bei dem nur Gruppen identifiziert werden, wird bei der modell-basierten Methodik zusätzlich eine charakteristische Beschreibung für die Gruppen gefunden. [19]

2.3.2 Erfahrung E

Unter der Erfahrung E versteht man den Prozess, bei dem Daten in einen Machine Learning Algorithmus eingespielt werden, sodass dieser mögliche Muster in den Daten erkennt und dadurch dazulernt. „Thus, the idea of a model or algorithm gaining experience usually occurs as an iterative process, also known as training the model.“ [18, S.12] Der Algorithmus erhält durch das Einspielen weiterer Daten Erfahrung. Dabei ist es nicht relevant, wie und wann man dem Algorithmus die Daten übergibt. Entweder können alte Daten auf einmal in den Algorithmus eingespielt werden, oder man spielt schrittweise neu erlangte Daten dazu. Der Machine Learning Algorithmus soll die Erfahrung E, die er durch das Lösen der Aufgabe T erlangt hat, zukünftig für unbekannte Datensätze anwenden können um Problemstellungen lösen zu können. [18]

2.3.3 Performance P

Angenommen, ein Machine Learning Algorithmus erfüllt eine Aufgabe T und sammelt dabei anhand von Datensätzen über eine gewisse Zeitspanne die Erfahrung E - wie kann man herausfinden, ob die Algorithmen sich den Vorstellungen entsprechend verhalten? Um diese Frage beantworten zu können, benötigt man eine Definition für die Performance P. Bei der Performance P handelt es sich meistens um eine Metrik, die feststellen soll, wie gut die Algorithmen die Aufgabe T, mit der gesammelten Erfahrung E, lösen.

Die Metriken werden normalerweise für jede Aufgabe T einzeln realisiert, jedoch gibt es auch Standardmetriken, die über Jahre hinweg entwickelt wurden und zum Einsatz kommen. [18]
Die Performancemessung wird, basierend auf der Übereinstimmung der tatsächlichen Objektklasse mit der Vorhersage des Algorithmus, durchgeführt. Dabei werden vier verschiedene Kategorien unterschieden (vgl. Abbildung 2.5). Die vier Klassen sind „True Positive“ (TP), „True Negative“ (TN), „False Positive“ (FP) und „False Negative“ (FN). Die Werte „True Positive“ und „False Negative“ sind die Objekte, wo die vorhergesagte Klasse mit der wirklichen Klasse übereinstimmt. Die anderen zwei Kategorien „True Negative“ und „False Positive“ sind jene, wo der vorhergesagte Wert nicht der tatsächlichen Klasse entspricht. [20]

		Predicted Class	
Actual Class		**Class = OK**	**Class = NOK**
	Class = OK	**True Positive**	**True Negative**
	Class = NOK	**False Positive**	**False Negative**

Abb. 2.5: Übereinstimmung tatsächliche Klasse - Vorhersage

Diese vier Typen sind die Grundlage für die verschiedenen Arten der Performancemessung:

- **Genauigkeit**: Diese Art der Performancebestimmung ist relativ einfach, da sie das Verhältnis aus allen richtig bestimmten Objekten zu der Gesamtheit der Objekte ist. [20]

$$\epsilon = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.1}$$

- **Präzision**: Die Präzision gibt das Verhältnis der korrekt vorhergesagten Werte zu allen positiv bestimmten Werten an. Dadurch kann bestimmt werden, wieviele Objekte unter allen positiv markierten tatsächlich korrekt sind. [20]

$$\rho = \frac{TP}{TP + FP} \tag{2.2}$$

- **Sensitivität/Trefferquote**: Mithilfe der Trefferquote werden die „True Positive“ Werte zu der Summe der „True Positive“ und der „False Negative“ in Verhältnis gesetzt. Dadurch kann bestimmt werden, wieviele von den richtigen Werten auch tatsächlich korrekt markiert werden. [20]

$$\pi = \frac{TP}{TP + FN} \tag{2.3}$$

- **Fehlerraten**: Die Fehlerrate oder auch „F1-Score“ ist der gewichtete Durchschnitt der Präzision und der Trefferquote. Die berechnete Fehlerrate berücksichtigt sowohl die „False Positive“ als auch die „False Negative“ Werte. [20]

$$F_1 = \frac{2\rho\pi}{\rho + \pi} \tag{2.4}$$

Die Berechnungen werden entweder mithilfe von Datensätzen durchgeführt, die der Algorithmus zum Trainieren bekommt oder mit vollkommen neuen Datensätzen. Der Sinn dahinter besteht darin, dass der Machine Learning Algorithmus, unabhängig von den Trainingsdaten, die er zur Lösung der Aufgabe T bekommt, auch für zukünftige neue Datensätze gut funktionsfähig ist.

Die Frage, wieviele dieser Kennwerte berechnet werden sollen, ist nicht einfach zu beantworten. Oft genügt es, entweder die Genauigkeit, die Fehlerrate oder die Präzision zu berechnen, jedoch kann es auch vorkommen, dass es schwieriger ist, die richtige Wahl zu treffen um die Performance des Machine Learning Algorithmus so genau wie möglich bestimmen zu können. [18]
Wie gut die Performance des Machine Learning Algorithmus ist, hängt zu einem Großteil auch davon ab, wie man die Features wählt, die aus den Daten extrahiert werden. [18]

2.4 Feature Engineering

Das Feature Engineering beschäftigt sich mit der Vorbereitung von Daten für Machine Learning Algorithmen. Eine gute Definition für Feature Engineering liefert Dr. Jason Brownlee, ein Softwareentwickler und Spezialist auf dem Gebiet künstliche Intelligenz:

„Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.“ [18, S.182]

Ein Beispiel für Feature Engineering ist die Generierung von Features aus der Pixelanzahl und den RGB-Farben von Bildern. [18]
Dabei wird diesem Bereich eine hohe Wichtigkeit zugesprochen, da die richtigen Features den Arbeitsaufwand für die Modellbildung reduzieren können und somit der gesamte Prozess eine höhere Erfolgschance hat. [10]
„Some people estimate that 80% of their effort in a machine learning application is spent on feature engineering and data cleaning.“ [10, S.10] Trotz der hohen Wichtigkeit des Feature Engineering wird die Thematik nur selten separat behandelt. Ein Grund hierfür könnte sein, dass die Features immer speziell für die Daten festgelegt werden und es schwierig ist, Verallgemeinerungen aus verschiedenen Aufgabenstellungen zu definieren. [10]

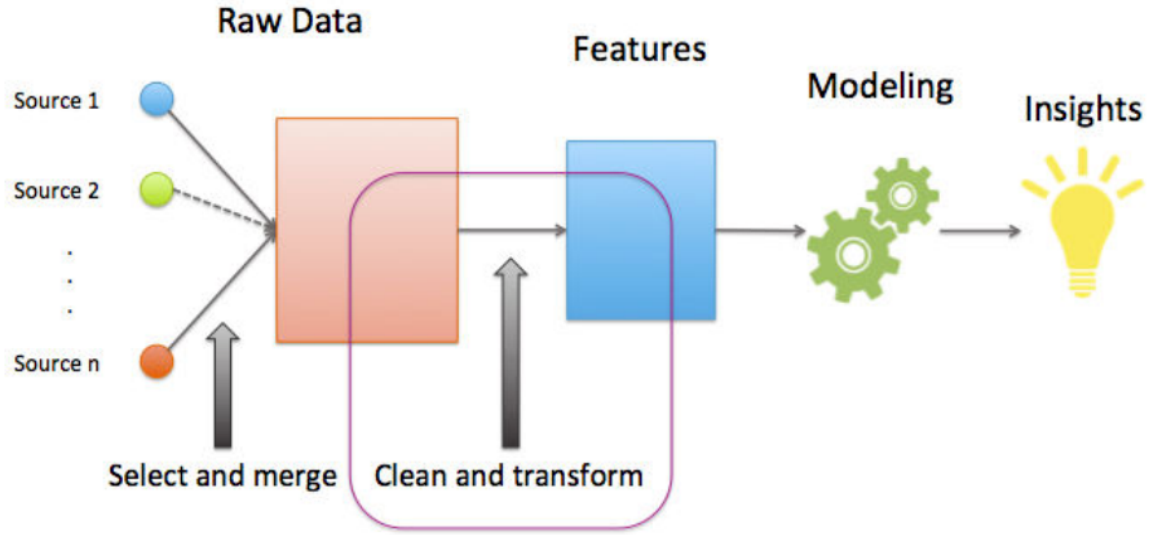

Abb. 2.6: Feature Engineering im Machine Learning Prozess [10]

Zwei wichtige Einheiten, die im Workflow einer Machine Learning Applikation und im Feature Engineering auftreten, sind Modelle und Features (siehe Abbildung 2.6). Die Modelle beschreiben Beziehungen zwischen unterschiedlichen Attributen der Daten. Numerische Daten können durch mathematische Formeln in Beziehung gesetzt werden, jedoch kommt es öfters vor, dass nicht numerische Daten vorliegen. [10]

„A feature is a numeric representation of raw data“ [10, S.13]. Die Auswahl der Features ist eng mit den Eigenschaften der Daten verbunden. [10] Es gibt zwei Arten von Features. Einerseits gibt es inhärente Features, die ohne weitere Veränderung der vorliegenden Daten entnommen werden und andererseits gibt es Features, die durch Feature Engineering aus den Eigenschaften der Daten generiert werden. [18]

Zudem besteht eine enge Verbindung der Features mit den Modellen. Im Prozess des Machine Learnings ist nicht nur die Wahl des Modells, sondern auch die Wahl der Features ausschlaggebend, denn die Wahl des einen beeinflusst die Wahl des anderen. Schlechte Features können das Modell komplizierter als nötig machen. Dieselbe Performance könnte mit anderen Features und einem wesentlich einfacheren Modell erreicht werden. Manche Features sind geeigneter für gewisse Modelle und umgekehrt. Daraus folgt, dass Feature Engineering der Prozess ist, um die geeignetsten Features bei gegebenem Modell und gegebenen Daten zu finden. [10]
Features sind davon abhängig, welche Problemstellung vorliegt. Auch wenn verschiedene Probleme ähnlich sind, können sich die Features sehr voneinander unterscheiden. [18]

Um die richtigen Features für ein Problem zu finden, sind oft mathematische Methoden, kombiniert mit Erfahrung und Intuition, notwendig. Beispiele für Features sind:

- Anzahl des in einem Textdokumentes vorkommenden Wortes
- Aufrufe eines Videos oder eines Liedes
- Bestimmung des Alters mittels des Geburtsdatums und dem heutigen Datum
- Informationen über die Pixel eines Bildes

Nicht nur die Auswahl der Features, sondern auch die Anzahl der Features ist ausschlaggebend. „Once we start extracting attributes or features from raw data samples, sometimes our feature space gets bloated up with a humongous number of features.“ [18, S.40]

In Machine-Learning Aufgaben, in denen „Unsupervised“ Methoden zur Anwendung kommen, kann es vorkommen, dass die Anzahl der Features sehr groß wird. Dadurch entsteht eine hohe Komplexität und es entstehen Probleme hinsichtlich des Trainings des Modells und andererseits hinsichtlich des Speichers. Diese Problematik wird auch „Fluch der Dimensionalität“ genannt. [18] Abhilfe schaffen hier Algorithmen, die in der Lage sind die Dimensionalität der Features zu verringern. Dabei wird in zwei Kategorien unterschieden:

- **Methode der Feature Selection:** Aus der ursprünglichen Menge der Features werden für jede Datenprobe spezifische Features ausgewählt. Dabei werden keine neuen Features generiert. [18]
- **Methode der Feature Extraction:** Aus den vorhandenen Features werden neue Features generiert. Das somit reduzierte Featureset besteht aus Features, die im ursprünglichen Featureset nicht enthalten waren. Ein bekanntes Verfahren aus der Statistik ist die Hauptkomponentenanalyse, bei der Datensätze derart strukturiert und vereinfacht werden, sodass so wenig Information wie möglich verloren geht. [18]

Für die Generierung der Features gibt es keine strikten Regeln. Es muss das Fachwissen und mathematische Operationen kombiniert werden, um aus den unbearbeiteten Daten die gewünschten Features bilden zu können. Es können verschiedene Datentypen verwendet werden, um Features zu generieren, wie zum Beispiel:

- Numerische Daten
- Text-Daten
- Bild-Daten

Feature Engineering ist somit ein wichtiger Teil des Machine Learning Workflows, der sowohl kleine Teile beeinflussen kann, wie beispielsweise die Modellbildung, aber auch die gesamte Applikation. Das Feature Engineering kann den Unterschied ausmachen, ob eine spezifische Problemstellung zufriedenstellend gelöst werden kann oder nicht. [18]

2.5 Ermittlung relevanter Zeitreihendaten

Um bedeutsame Zeitreihendaten identifizieren zu können und dem/der User/in zu präsentieren, kommen in der Literatur unterschiedliche Methoden zur Anwendung. Im nächsten Abschnitt werden Methoden und Techniken erläutert, die die Methodik in dieser Arbeit maßgeblich beeinflusst haben.

2.5.1 Relevance User Feedback in Zeitreihendaten

Neben statistischen Auswertungen, wie der Bestimmung von Korrelationen und Trends, ist die visuelle Darstellung von Zeitreihendaten für die UserInnen von hoher Bedeutung. Zeitreihendaten werden in vielen unterschiedlichen Bereichen, wie beispielsweise Medizin, Wirtschaft und in den Ingenieurwissenschaften gespeichert und analysiert. [8] Beispiele für Anwendungen sind:

- Medizin: Darstellung der EKG Daten um Rythmusstörungen des Herzens zu erkennen [8]
- Wirtschaft: Analyse von Aktienkursen um Trends für die Zukunft zu erkennen [8]
- Ingenieurwissenschaften: Analyse von Zeitreihendaten aus Bohrprozessen [21]

Das Problem ist, dass aufgrund der großen Menge an Daten die UserInnen immer nur einen Bruchteil begutachten können. Der/die User/in weiß im Vorhinein oft gar nicht genau was er/sie sucht und als relevant empfindet. Zusätzlich ist die Einstufung einzelner Abschnitte subjektiv - die Meinung zweier UserInnen bezüglich eines Zeitreihenausschnittes kann unterschiedlich sein. [8]
Diese Probleme haben zur Methodik des „Relevance Feedback“ geführt. „Relevance feedback retrieval systems prompt the user for feedback on retrieval results and then utilize this feedback on subsequent retrievals with the goal of increased retrieval performance.“ [22, S.1] Systeme, die mit „Relevance Feedback“ arbeiten, fordern die UserInnen auf, zu gewissen Resultaten Feedback zu geben, damit dieses an das System zurückgegeben werden kann. [22]
In der Arbeit von MacArthur et al. [22] wird ein solches „Relevance Feedback“ System

entwickelt. Das entwickelte System erfüllt folgende Anforderungen:

- Das System benötigt eine begrenzte Anzahl an Feedback. Dadurch wird die investierte Zeit der UserInnen minimiert [22]
- Das System benötigt ein oder zwei Iterationen um gute Resultate zu generieren [22]
- Das System ist schnell genug für eine online Applikation [22]

Den UserInnen werden Bilder präsentiert, die diese als relevant oder irrelevant einstufen. Die Feedback Information wird anschließend in einem Decision Tree Algorithmus ausgewertet. Der Algorithmus arbeitet nicht direkt mit den Bildern, sondern mit den Features, die mit diesen assoziiert werden. Die Features werden mithilfe einer Software generiert. Das Training des Machine Learning Algorithmus hat zum Ziel, die Bilder identifizieren zu können, die die UserIn am meisten interessieren. [22]

2.5.2 Ansätze zur Identifizierung relevanter Zeitreihen

Die Information, die aus Zeitreihendaten abgelesen werden kann, ist nicht nur in den Ingenieurwissenschaften, sondern auch in den Bereichen Wirtschaft und Medizin von Interesse. [9]
Anfang der 2000 Jahre hat es intensive Forschungsarbeit auf dem Gebiet der Zeitreihen gegeben. In einer Arbeit von Eamonn Keogh [9], einem amerikanischen Informatiker, der sich mit der Auswertung von Zeitreihendaten beschäftigt, wird ein Ansatz zur effizienten Klassifizierung entwickelt. Aufgrund der sehr großen Datenmengen waren Machine Learning Algorithmen in der Klassifizierung und Gruppierung von Zeitreihen nur bedingt erfolgreich. Die Schwierigkeit ist, einerseits das Problem der hohen Dimensionalität[1] zu lösen, und andererseits eine geeignete Methode zu finden, um Ähnlichkeiten zwischen den Bereichen feststellen zu können. Für die Überprüfung einzelner Abschnitte werden die Datenreihen in einzelne Segmente unterteilt, um spezielle Abschnitte identifizieren zu können. Zusätzlich werden die Zeitreihenabschnitte mit einem Gewichtsvektor versehen. Die Gewichtung wird durch die UserInnen festgelegt. Diese können die Abschnitte als wichtig oder unwichtig einstufen. [9]

[1] für Daten mit hoher Dimensionalität gilt: $p > n$, p...Anzahl Features/Datenpunkt, n...Anzahl an Datenpunkten [19]

In einer weiteren Arbeit hat Keogh u.a. die Frage nach den Abgrenzungen von Zeitreihenabschnitten behandelt. [7] Ausgangspunkt dieser Arbeit ist folgende Annahme:

„Most literature on time series classification assumes that the beginning and ending points of the pattern of interest can be correctly identified, both during the training phase and later deployment. “ [7, S.1]

Diese ungerechtfertigte Annahme wird in dieser Arbeit verworfen und es wird gezeigt, dass die korrekte Identifizierung schwieriger ist, als bisher angenommen. Das Ziel dieser Arbeit ist es, ein Framework zu erstellen, um die Problematik der Identifizierung zu erleichtern. “The framework requires only very weakly annotated data, such as “in this ten minutes of data, we see mostly normal heartbeats...,”„ [7, S.1].
Folgende Annahmen, die in der Literatur angenommen werden, werden in dieser Arbeit verworfen:

- Es können *viele fast idente* Abschnitten beobachtet werden
- Die Abschnitte haben alle *dieselbe Länge*
- Jedes Objekt, das klassifiziert wird, kann *eindeutig* in eine vordefinierte Gruppe zugeordnet werden

Für die Erstellung des Frameworks wird ein Datenobjekt namens „Data Dictionary“ kreiert. Diese Objekte beinhalten eine „intelligente“ Teilmenge der Trainingsdaten. Der einzige Parameter, der verwendet wird, ist der prozentuelle Anteil der Trainingsdaten, der diesen Abschnitt ausmacht. Nach Tests mithilfe realer Daten wurde gezeigt, dass die Klassifizierung schneller und genauer arbeitet als Algorithmen, die eine genaue Beschreibung der Daten benötigen. [7]

Ein weiterer wichtiger Gesichtspunkt im Data Mining von Monitoringdatenreihen ist das Erkennen von Ereignissen, auch „Event Detection“ genannt. [23] Dabei geht es um die Auffindung von speziellen Punkten, die für die UserInnen von Bedeutung sind. „These events often are points in a time series that can be peaks, level changes, sudden changes of spectral characteristics, etc.“ [13, S.1] Diese Punkte werden im Vorhinein markiert, um einen Machine Learning Algorithmus zu trainieren. Arbeiten auf diesem

Gebiet versuchen Algorithmen zu entwickeln, die automatisch solche „Events“ auf Basis der Trainingsdaten, auffinden können. [13]

3 Aufbau und Beschreibung der Applikation

Im Folgenden wird mittels eines Top-Down Ansatzes die Applikation erklärt. Ausgehend von der verwendeten Programmiersprache wird anfangs die prinzipielle Struktur und die Methodik erklärt. Des Weiteren wird die Auswahl der Features beschrieben. Darüber hinaus wird die Integration und die Implementierung des Machine Learning Programmabschnittes erläutert. Die Struktur des Programms, das zur Problemlösung erstellt wird, ist in Abbildung 3.1 dargestellt.

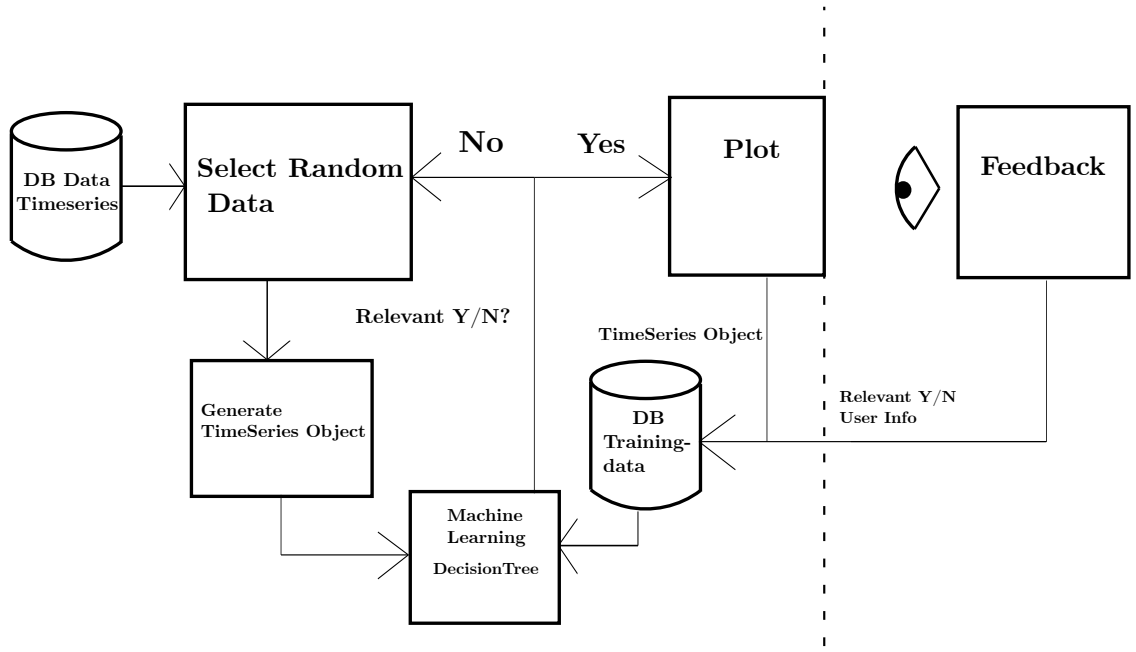

Abb. 3.1: Struktur der Machine Learning Applikation

3.1 Methodik und Struktur

Für die Erstellung der Softwareapplikation wird die Programmiersprache Python verwendet. Im Speziellen wird ein effizientes Python Web Framework namens „Django“ verwendet. Der große Vorteil von Django ist, dass das Web Development und die Python Programme sehr gut kombiniert werden können. [24] Zur Erstellung des GUI wird die Programmiersprache HTML[1] in Kombination mit CSS[2] verwendet.

Die Applikation muss folgende Anforderungen erfüllen:

- Die UserInnen müssen interaktiv entscheiden können, ob die dargestellte Datenreihe als relevant oder irrelevant klassifiziert werden.
- Die dargestellten Plots sollen durch den Machine Learning Algorithmus vorab überprüft werden, ob diese potentiell für die UserInnen relevant sind.
- Das Training des Machine Learning Algorithmus soll zeitgleich erfolgen. Sobald die UserInnen Entscheidungen treffen, wird die Metainformation an den Machine Learning Algorithmus übergeben.
- Die Applikation soll im Web funktionieren, ohne dass die Usability durch lange Ladezeiten beeinträchtigt wird.

Für die Applikation werden zwei Datenbanken verwendet. Die erste Datenbank beinhaltet die Monitoringdaten, die zur Erstellung der Plots verwendet werden. Die zweite Datenbank wird zum Speichern und zur Verwaltung der Metainformationen verwendet. Das verwendete Datenbanksystem ist SQLite. Dieses System wird gewählt, weil es ohne Konfiguration und Server auskommt. Außerdem zeichnet es sich durch schnelle Schreib- und Lesegeschwindigkeit, sowie durch eine geringe Speichergröße aus. [25]
Zur Generierung der Plots werden zufällig Daten aus der ersten Datenbank selektiert und daraus die Plots und Features erstellt. Diese Plots werden den UserInnen präsentiert und diese entscheiden anschließend, ob sie die Plots als relevant oder irrelevant einstufen. Diese Methodik wird als „Relevance Feedback Retrieval“ bezeichnet. [8] Da man a priori

[1] Hypertext Markup Language
[2] Cascading Style Sheets

nicht wissen kann, welche Zeitreihen die UserInnen als relevant oder irrelevant einstufen, werden diesen auf Basis von zufällig gewählten Datenreihen Plots angezeigt, die von den UserInnen klassifiziert werden. Je nach Erfahrung und Wissen über die Daten werden die Plots unterschiedlich bewertet.

Im GUI der Applikation werden drei Plots dargestellt und können separat durch Buttons als relevant oder irrelevant eingestuft werden (siehe Abbildung 3.2) Nachdem die UserInnen einen der Buttons gedrückt haben, bekommen diese selbst ein Feedback. Dieses soll zum einen dazu dienen, die Eingabe zu bestätigen und andererseits sollen die UserInnen dadurch gehindert werden, den Plot nochmal einstufen zu können. Die Bestätigung ersetzt beide Buttons. (siehe Abbildung 3.3)

Abb. 3.2: Screenshot des GUI: Fenster zur Klassifikation der Plots

Durch die Klassifizierung der UserInnen werden die Labels „relevant“ oder „irrelevant“ entsprechend erstellt und abgespeichert. Die durch das UserInnen-Feedback generierten Daten werden anschließend als Trainingsdaten für den Machine Learning Algorithmus verwendet. Nachdem die UserInnen alle drei Plots bewertet haben, betätigen sie den „Refresh“ Button um drei neue Plots zu generieren. Um Feedback vom Machine Learning Algorithmus zu erhalten, wird das jeweilige Feature auf Relevanz überprüft. Werden diese als relevant eingestuft, wird anschließend der Plot generiert. Werden diese hingegen als irrelevant klassifiziert, werden neue Daten aus der Datenbank selektiert und die Features erneut überprüft.

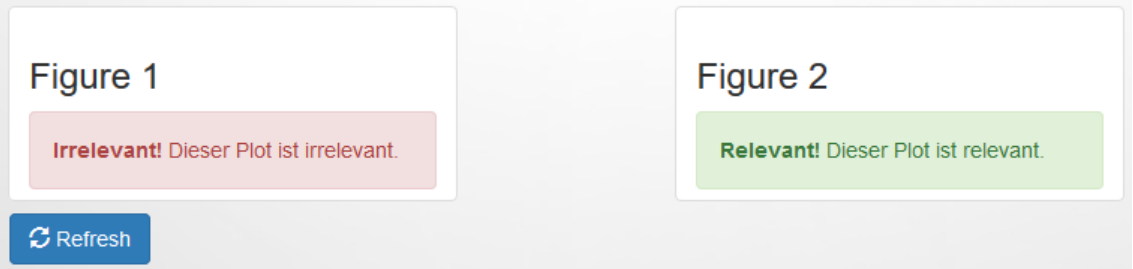

Abb. 3.3: Feedback nach Betätigen der Buttons

Um die Metainformationen verarbeiten zu können, werden diese in der zweiten Datenbank abgespeichert. Um das Verhalten des Algorithmus auf unterschiedliche Gruppen von Features zu untersuchen, wird im nächsten Abschnitt die Wahl der Features sowie die Wahl der Machine Learning Methodik näher beschrieben.

3.2 Auswahl der Features

Der Machine Learning Algorithmus wird mit drei verschiedenen Featuresets trainiert. Unter einem Featureset versteht man eine Gruppe von Features. Wie in Kapitel 1 beschrieben, gibt es zwei Gruppen: inhärente und generierte Features. Aus diesem Grund besteht ein Featureset aus inhärenten Features und für ein weiteres werden die Features mithilfe eines mathematischen Modells erstellt. [18] Das dritte Featureset stellt eine Kombination aus beiden vorigen dar. Alle drei Featuresets haben zwei zusätzliche Features. Es wird einerseits die Zeitspanne `span` zwischen Anfang und Ende, und andererseits der Name des Sensors `unit` hinzugefügt. Da physikalische Werte immer eine Einheit haben und diese in der Datenbank nicht verfügbar ist, wird der Name des Sensors herangezogen.
Um den Algorithmus mit den unterschiedlichen Featuresets trainieren zu können müssen sich die UserInnen für einen der drei farbigen Buttons entscheiden. Hinter jedem dieser Buttons werden andere Features zum Trainieren des Machine Learning Algorithmus eingesetzt (siehe Abbildung 3.4). Die Farben werden wiefolgt den Featuresets zugeordnet: Der grünen Farbe sind die inhärenten Features zugeordnet, der roten Farbe sind die generierten Features zugeordnet und der blauen Farbe sind die kombinierten Features zugeordnet.

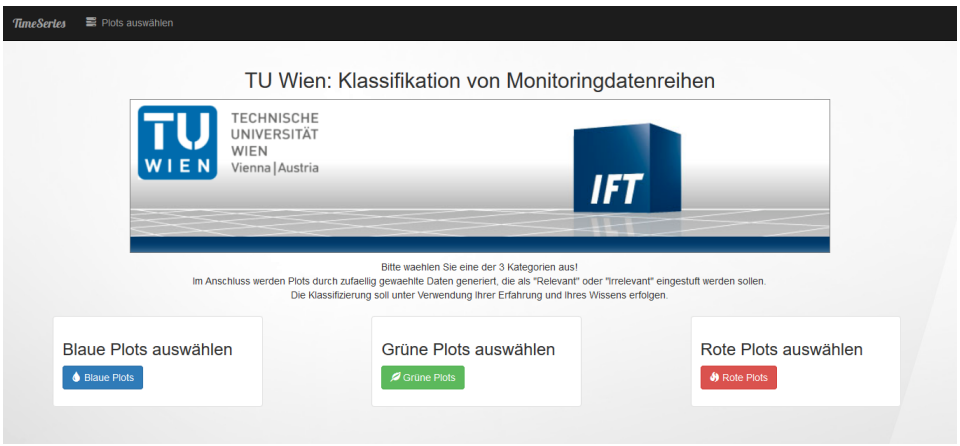

Abb. 3.4: Screenshot des GUI: Auswahlfenster für unterschiedliche Featuresets

In den nachfolgenden Unterkapitel werden die einzelnen Featuresets und deren Aufbau detailliert beschrieben.

3.2.1 Inhärente Features (grüne Features)

Das erste Featureset beinhaltet Informationen, die von den Daten direkt entnommen werden. Diese Art von Features wurde bereits von Jason Brownlee [26] bei der Bearbeitung von Zeitreihendaten herangezogen. Bei den untersuchten Datenreihen von Brownlee handelte es sich um Temperaturdaten. Der Machine Learning Algorithmus wird bei der Auswahl des grünen Buttons mit diesen Features trainiert. (siehe Abb.3.4)
Die Verfügbarkeit für inhärente Features ist beschränkt. Es können beispielsweise das Monat, der Tag oder die Stunde des Zeitstempels oder spezielle numerische Werte der Messung herangezogen werden. Die ersten beiden Features werden aus den Informationen vom ersten Eintrag und vom letzten Eintrag entnommen. Hierfür wird der Sekundeneintrag des Zeitstempels herangezogen. [26]
Um auch Informationen aus den Messwerten heranzuziehen, werden als drittes und viertes Feature der minimale und maximale Wert der Datenreihe herangezogen. In Arbeiten von Valery Guralnik und Jaideep Srivastava [23] und André Gensler und Bernhard Sick [13] wurden Zeitreihendaten hinsichtlich Punkten bzw. „Events“ untersucht, die interessante Abschnitte definieren. Diese zwei Werte wurden zudem von Brownlee herangezogen. [26] Um diese Information aus der Datenreihe zu extrahieren, wird das Python Package `numpy` verwendet. Dieses beinhaltet eine Funktion, um das Minimum bzw. das Maximum einer Datenreihe zu bestimmen.

Tab. 3.1: Darstellung des inhärenten Featuresets

TimeSeries Green
start second
end second
minimum
maximum
unit
span

Um die Features übersichtlich aufzulisten, wird das ganze Featureset in Tabelle 3.1 dargestellt.

3.2.2 Generierte Features (rote Features)

Das zweite Featureset wird durch Modellierung der Datenreihe bestimmt. Hierfür werden Gleichungen für die Berechnung statistischer Kennwerte herangezogen.
Die vier Features, die berechnet werden, sind die statistischen Kennwerte Mittelwert μ, Standardabweichnung σ, Schiefe ν (engl. skewness) und Wölbung ω (Kurtose - engl. kurtosis). [27] Die Schiefe und Kurtose geben Auskunft über die Form der Verteilung der Zeitreihendaten. Im Speziellen charakterisiert die Schiefe ν den Grad der Asymmetrie um den Mittelwert. Abbildung 3.5 stellt eine links- ($\nu > 0$) und rechtssteile Kurve ($\nu < 0$) dar. [28]

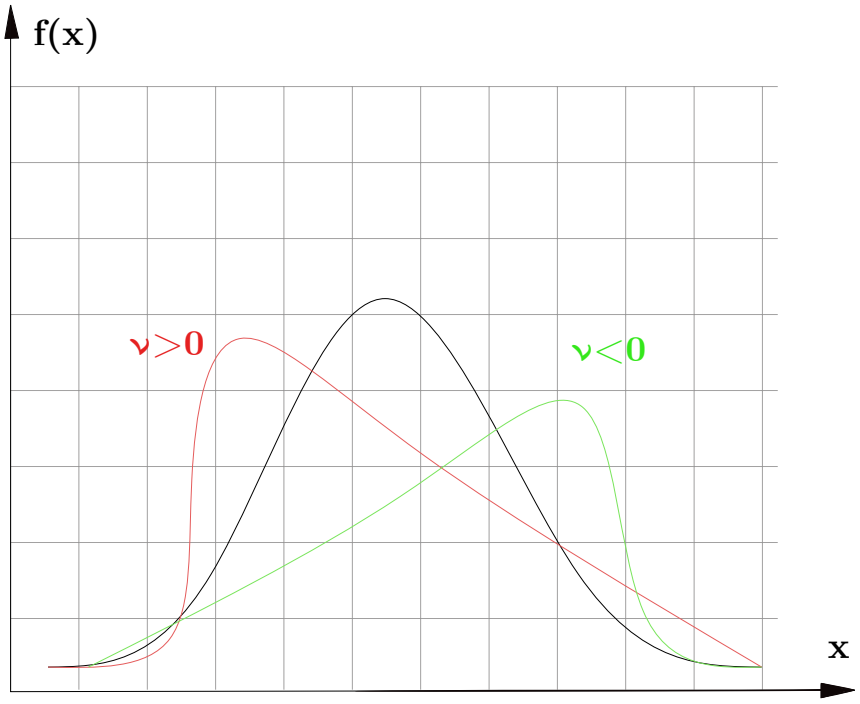

Abb. 3.5: Darstellung der statistischen Größe „Schiefe“

Die Kurtose ω vergleicht die Flachheit oder Spitzheit der Werteverteilung relativ zur Normalverteilung. [27] Abbildung 3.6 stellt einerseits eine leptokurtische Kurve (spitz - $\omega > 3$) und andererseits eine platykurtische Kurve (flach - $\omega < 3$) dar. Für die Normalverteilung gilt $\omega = 3$ (mesokurtische Kurve). [28]

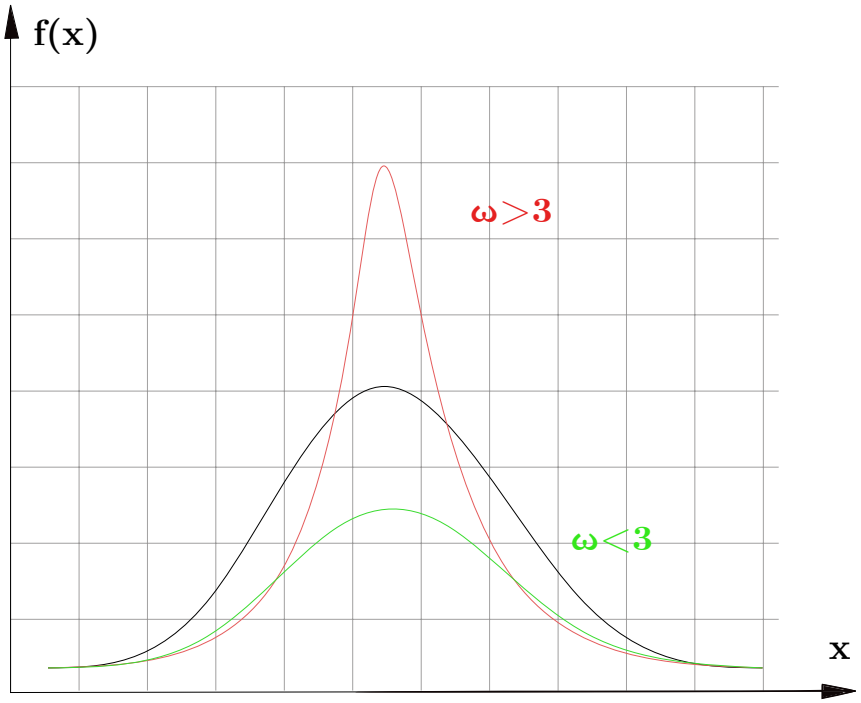

Abb. 3.6: Darstellung der statistischen Größe „Wölbung“

Die Auswahl dieser Features basiert auf der Arbeit von Alex Nanopoulos u.a. [27], die sich mit der Klassifikation von Zeitreihendaten mithilfe eines neuronalen Netzwerkes beschäftigt. Die verwendeten Daten stammen aus der statistischen Prozesslenkung. In dieser Arbeit wird gezeigt, dass die feature-basierte Methode effizienter als jene arbeitet, die die Klassifikation auf Grundlage der Werte der Zeitreihe durchführt und auch bei langen Zeitreihen gut funktioniert. Die verwendeten Formeln zur Berechnung der Features lauten:

Mittelwert μ:

Der Mittelwert gibt die Lage von Daten an. Er entspricht der Summe der einzelnen Daten, dividiert durch die Gesamtanzahl der Daten. [6]

$$\mu = \frac{\sum_{t=1}^{n} y(t)}{n} \tag{3.1}$$

Standardabweichung σ:

Die Standardabweichung gibt die Streuung der Daten an. Berechnet wird sie aus der

Wurzel der Varianz. Dabei entspricht die Varianz der Summe der Quadrate aus der Differenz der Daten und Mittelwert, dividiert durch die Gesamtanzahl der Daten. [6]

$$\sigma = \sqrt{\frac{\sum_{t=1}^{n}(y(t)-\mu)^2}{n}} \tag{3.2}$$

Schiefe ν:

Die Schiefe ist ein Maß für die Symmetrie oder Asymmetrie einer Verteilung. Sie gibt an ob die Verteilung symmetrisch, rechtssteil oder linkssteil ist. [28]

$$v = \frac{\sum_{t=1}^{n}(y(t)-\mu)^3}{n\sigma^3} \tag{3.3}$$

Wölbung ω:

Die Wölbung, oder auch Kurtose genannt, ist ein Maß für die Krümmung einer Verteilung. Eine kleine Wölbung bedeutet eine flachere Verteilung, wohingegen eine große Wölbung eine spitzere Verteilung bedeutet. [28]

$$w = \frac{\sum_{t=1}^{n}(y(t)-\mu)^4}{n\sigma^4} - 3 \tag{3.4}$$

Das Modell zur Berechnung der Features wurde mithilfe des Python Packages `numpy` erstellt. Dieses Package beinhaltet bereits implementierte Funktionen zur Berechnung der statistischen Kennwerte. Der Vorteil ist, dass die bereitgestellten Funktionen effizienter arbeiten, als jene, die neu implementiert werden müssen.

Tab. 3.2: Darstellung des generierten Featuresets

TimeSeries Red
mean
std
skew
kurt
unit
span

Zur übersichtlicheren Abbildung der Features sind diese in Tabelle 3.2 dargestellt.

3.2.3 Kombinierte Features (blaue Features)

Das dritte Featureset stellt eine Kombination aus Features aus den inhärenten und den generierten Features dar. Dabei werden einerseits Informationen über das Maximum und das Minimum aus den Datenreihen übernommen, und andererseits werden der Mittelwert μ und die Standardabweichung σ berechnet. Die kombinierten Features sollen eine mögliche Korrelation der Features und deren Auswirkung auf den Lernerfolg des Machine Learning Algorithmus zeigen. Die Auswahl der vier Features erfolgte zufällig.
Das Featureset, das die kombinierten Features enthält, wird in Tabelle 3.3 abgebildet.

Tab. 3.3: Darstellung des kombinierten Featuresets

TimeSeries Blue
minimum
maximum
mean
std
unit
span

3.3 Auswahl der Machine Learning Methodik

Um eine passende Machine Learning Methodik auszuwählen, sollen folgende Kriterien erfüllt sein:

- Die Methodik soll einfach sein
- Die Featuresets können dem Algorithmus übergeben werden
- Der Algorithmus kann anhand der Featuresets eine Vorhersage treffen, ob die Datenreihen potentiell relevant oder irrelevant für die UserInnen sind
- Das Training mit wenigen Datenobjekten soll möglich sein

Dabei stehen Algorithmen aus folgenden drei Gruppen zur Verfügung:

- Supervised
- Unsupervised
- Deep Learning

„More sophisticated learners are seductive, but they are usually harder to use, because they have more knobs you need to turn to get good results, and because their internals are more opaque.“ [29, S.6] Aufgrund der komplexen Struktur und der automatischen Generierung der Features im Deep Learning kommt diese Methodik nicht zum Einsatz.

Um Problemstellungen zu lösen, bei denen aufgrund von Trainingsdaten ein Algorithmus Datenobjekte einer Gruppe zuordnet, eignet sich die Supervised Methodik. Die Datenobjekte sind jene Objekte, die im Programm die Featuresets repräsentieren. Beim Training von Algorithmen aus der Supervised Gruppe wird ein neues Datenobjekt erzeugt und dem Machine Learning Algorithmus übergeben. Dieser wird durch Datensätze, die bereits ein „Label“ besitzen, trainiert, damit dieser das neue Datenobjekt klassifizieren kann. Illustriert wird der Vorgang des Trainings in Abbildung 3.7.

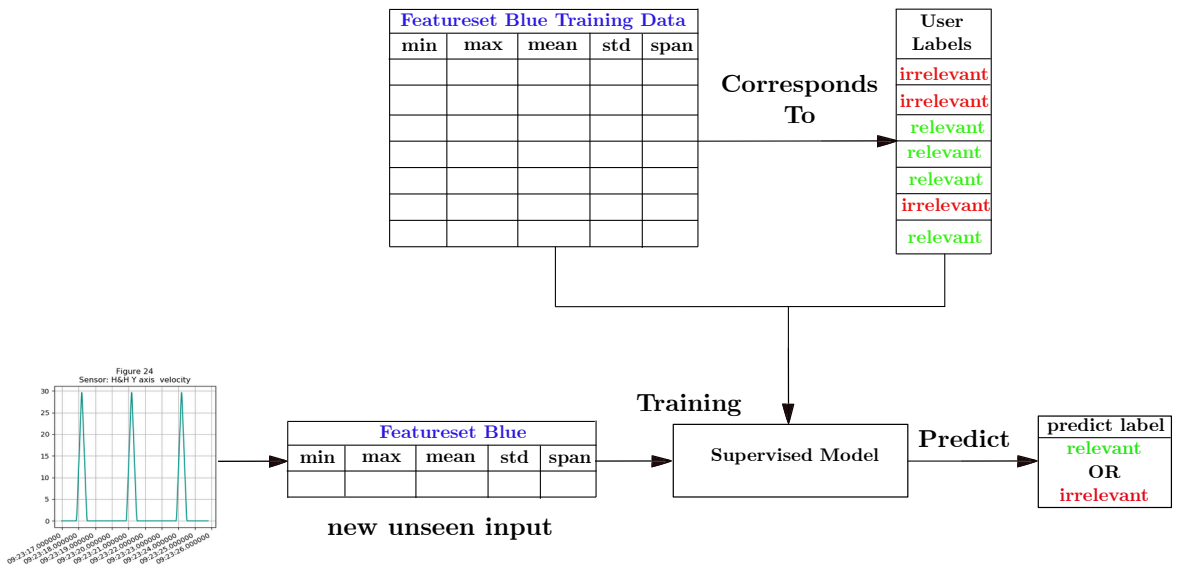

Abb. 3.7: Darstellung des Trainings (Bsp.: Featureset Blue)

Da anfangs noch keine Trainingsdaten vorhanden sind und diese erst durch die UserInnen erstellt werden müssen, muss die Anzahl der zu generierenden Datensätze betrachtet werden, um einen Klassifizierer mit hoher Genauigkeit erstellen zu können. In einer Arbeit von Figueroa u.a. [30] wird eine Methode entwickelt, um vorherzusagen, wie viele Datensätze benötigt werden, damit gute Klassifizierungsmodelle gebildet werden können. Die Vorhersage der Datenmenge wird auf Basis von Lernkurven gemacht, die vorab durch eine kleine Datenmenge erstellt wurden. „Depending on the data set and sampling method, it took between 80 to 560 annotated samples to achieve mean average and root mean squared error below 0.01“ [30, S.1]
Aufgrund der geringen Komplexität des Problems orientiert sich die Menge an Trainingsdaten an der unteren Grenze von 80 Datensätzen und wird mit 150 festgelegt.

3.4 Implementierung des Machine Learning Segments

Den Kern der Applikation stellt die Einbindung des Machine Learning Algorithmus dar. Dabei müssen ein paar Problemstellungen gelöst werden. Einerseits soll die Applikation bzw. der Machine Learning Algorithmus das generierte „TimeSeries“ Datenobjekt dahingehend überprüfen, ob dieses als relevant oder irrelevant eingestuft wird. Andererseits muss der Algorithmus immer mit den Informationen aus dem Feedback versorgt werden.

Durch die Betätigung eines farbigen Buttons durch die UserInnen wird eine Kette an Funktionen und Skripte ausgeführt (siehe Abbildung 3.8). Die aufgerufene URL wird zuerst im Skript `urls.py` passend zugeordnet. Nach der erfolgreichen Identifizierung der Seite wird in `views.py` die passende Funktion `monitoring` ausgeführt. Um die drei unterschiedlichen Farben der Featuresets unterscheiden zu können, wird durch die Betätigung des Buttons eine der Farben „blue“, „green“ oder „red“ übergeben.

Im nächsten Schritt öffnet das Programm die Funktion `get_data`. In der Subroutine `create_ts` wird eine zufällige Menge an Zeitreihendaten aus der Datenbank ausgelesen und je nach Farbe das passende „TimeSeries“ Objekt generiert. Die Objekte sind im Skript `timeseries.py` definiert. Dadurch ist es möglich, diese auf einfache Weise zu erstellen und anschließend darauf zugreifen zu können. Für die Überprüfung auf Relevanz wird das „TimeSeries“ Objekt nach Erstellung dem Skript `predict_me.py` übergeben. Dieses greift zuerst auf die zweite Datenbank zu, um die dort gespeicherten Daten auszulesen.

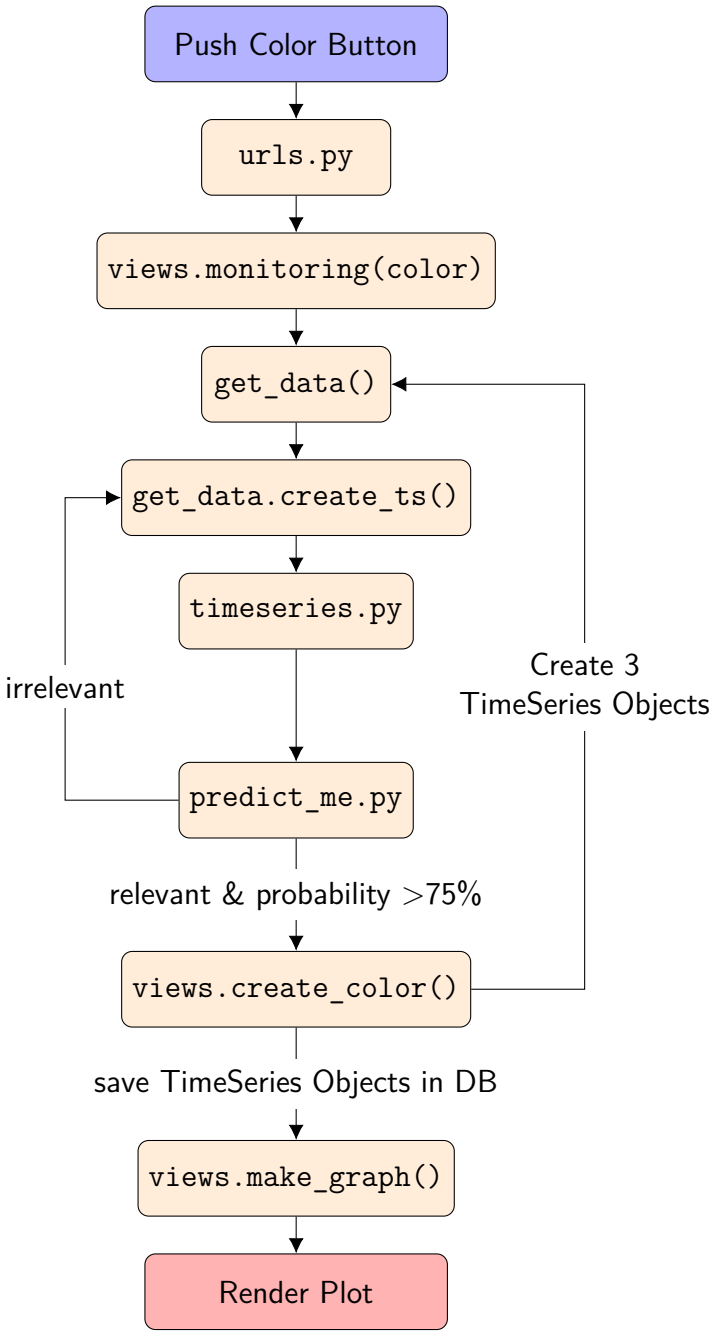

Abb. 3.8: Pogrammablauf: Erstellung der TimeSeries Objekte

Da anfangs keine Trainingsdaten vorhanden sind, wird ein alternativer Weg zum Machine Learning implementiert. Hierfür wird die Anzahl der vorhandenen Objekte abgefragt. Liegt diese unter drei wird das Objekt mit einer Wahrscheinlichkeit von 99 % als relevant eingestuft. (siehe Abbildung 3.9)

```
1     temp = df.shape[0] < 3

3     if temp is True:
          predict = 1
5         prob_rel = 0.99

7         return predict, prob_rel
```

Abb. 3.9: Abfrage Anzahl der Trainingsdaten

Sind mehr als drei Objekte in der Datenbank - das entspricht einem Durchgang des Programmes - wird das „TimeSeries“ Objekt durch den Machine Learning Algorithmus überprüft.

Aufgrund des Features `unit`, das den Sensornamen als `string` abgespeichert beinhaltet, sind weitere Bearbeitungsschritte notwendig. Der Sensorname muss in einen numerischen Wert transformiert werden, um ihn für den Machine Learning Algorithmus verwertbar zu machen. [31] Die Übersetzung der Bezeichnungen erfolgt mithilfe der Variable `trans`. Diese Variable wird in einem Python - `dictionary` abgespeichert, in dem jedem Sensornamen eine fortlaufende Zahl zugewiesen wird. Die Variable `trans` wird der Funktion `predict_me` als Input Variable übergeben. Die Übersetzung erfolgt in der Funktion `handle_non_numerical_data`. Diese Funktion übersetzt spaltenweise die Werte der übergegebenen Tabelle `frame`, falls diese nicht vom Typ `int` oder `float` sind. Dabei wird mit der Funktion `map` gearbeitet. Mithilfe von `map` kann auf mehrere Werte eine Funktion angewendet werden. Durch das Transformieren von Wörtern in numerische Werte ist es möglich, `strings` als Features einzusetzen. (siehe Abbildung 3.10)

```
1       def handle_non_numerical_data(frame, trans):
            columns = frame.columns.values
3
            for column in columns:
5
                def convert_to_int(val):
7                   return trans[val]

9               if frame[column].dtype != np.int64 and\
                        frame[column].dtype != np.float64:
11                  frame[column] = list(map(convert_to_int,
                                             frame[column]))
13          return frame
```

Abb. 3.10: Funktion: `handle_non_numerical_data`

Der verwendete Machine Learning Algorithmus ist der „DecisionTree“ aus dem Python-Package `sklearn`. Dabei handelt es sich um einen Algorithmus nach der Supervised Gruppe. Der DecisionTree wird für diese Applikation gewählt, weil dieser in der Arbeit von MacArthur u.a [22] zur Anwendung gekommen ist und dieser Algorithmus zudem zwei große Vorteile bietet:

- Einfachheit: Gegenüber Deep Learning Algorithmen benötigt der „DecisionTree“ weniger Einarbeitungszeit, um diesen in einer Applikation verwenden zu können.
- Verständlichkeit: Anhand der erzeugten Äste können die „DecisionTree“ Ergebnisse bzw. die Verzweigungen leicht interpretiert werden. Außerdem können die Verzweigungen grafisch dargestellt werden, um das Ergebnis interpretieren zu können.

Die Funktionsweise ist wie folgt: Der „DecisionTree“ startet bei einem spezifischen Feature und bildet zwei Äste. Bei den neu gebildeten Knoten werden anschließend anhand der verbleibenden Features weitere Verzweigungen gebildet. Diese Unterteilung erfolgt solange, bis die Zielvariable verfügbar ist. [18]
Zudem können verschiedene Parameter zusätzlich verändert werden. Beispiele für veränderbare Parameter sind die maximale Anzahl an Features (`max_features`) und die Tiefe des Baumes (`max_depth`). [18]

Im nächsten Schritt werden dem Machine Learning Algorithmus die Trainingsdaten übergeben. Dabei wird die Spalte mit der Relevanz-Information separat übergeben, da-

mit die Abfrage des TimeSeries Objektes auf Relevanz erfolgen kann. In Abbildung 3.11 ist der dazugehörige Codeausschnitt dargestellt.

```
        df = handle_non_numerical_data(df, trans)
        x = np.array(df.drop(['relevance'], 1))
        y = np.array(df['relevance'])

        clf = tree.DecisionTreeClassifier()
        clf.fit(X=x, y=y)
```

Abb. 3.11: Training des DecisionTree-Algorithmus

Dabei bedeutet numerisch gesehen „0“ irrelevant und „1“ relevant. Bei Übergabe des Datenobjektes wird vom Algorithmus zusätzlich bestimmt, mit welcher Wahrscheinlichkeit das Objekt als relevant eingestuft wird.

Bevor die Daten visualisiert werden, wird der Output des Machine Learnings überprüft. Folgende drei Abfragen werden gemacht: Relevanz ,Wahrscheinlichkeit und Anzahl der Versuche. Falls das Objekt als relevant eingestuft wird, demnach die Variable `predict` gleich 1 ist, muss dieses zusätzlich eine Wahrscheinlichkeit größer als 75% aufweisen, damit der dazugehörige Plot den UserInnen angezeigt wird. Die numerischen Werte für die Versuche sowie für die Wahrscheinlichkeit wurden beliebig gewählt. Um keine Endlosschleife zu erzeugen, werden sieben Versuche als Grenze festgelegt. Im letzten Durchgang wird automatisch das „TimeSeries“ Objekt zur Erstellung des Plots verwendet, unabhängig davon, ob dieses als relevant eingestuft wird oder nicht. (siehe Abbildung 3.12)

```
    ts, table, k, n, trans = create_ts(color)

    j = 0
    c = True
    while c is True:
        predict, prob_rel = predict_me(ts, trans, color)
        check = prob_rel > 0.75
        loop = j <= 6
        if loop is True:
            if predict == 1:
                if check is True:
                    c = False
                else:
                    ts, table, k, n, trans = create_ts(color)
                    j += 1
                    c = True
            else:
                ts, table, k, n, trans = create_ts(color)
                j += 1
                c = True
        else:
            c = False
```

Abb. 3.12: Überprüfungsmechanismus in der Funktion `get_data`

Im Anschluss der Überprüfung werden die Datenobjekte in der Funktion `create_color` umgewandelt, um sie in die Form zu bringen, die in der zweiten Datenbank abgespeichert werden kann. Dabei ist das Wort „color“ Platzhalter für die drei verfügbaren Farben. Dieser Schritt ist notwendig, da das Framework „Django“ nach diesen Vorgaben die Datenbank aufsetzt.

Nach diesem Schritt wird der Plot anhand der Funktion `make_graph` erstellt. Mithilfe von einem weiteren Datenobjekt „FigureData“, das mit dem dazugehörigen „TimeSeries“ Objekt verknüpft ist und alle notwendigen Informationen für die Erstellung der Grafik beinhaltet, werden erneut die Zeitreihendaten aus der ersten Datenbank ausgelesen. Dieser Schritt ist notwendig, da es nicht möglich ist, die Daten bei der Erstellung des „TimeSeries“ Objektes abzuspeichern. Durch die Verwendung der SQLite Datenbank erfolgen die Zugriffe auf die Datenbank schnell und die Usability wird dadurch nicht beeinträchtigt.

Schlussendlich wird der erzeugte Plot in der passenden HTML-Datei dargestellt und die

UserInnen können folglich das Bild bewerten.

Durch die Klassifikation der Plots durch den passenden Klick der Buttons (vgl. Abbildung 3.2) müssen die TimeSeries Objekte, die in der Datenbank liegen manipuliert werden. Abbildung 3.13 visualisiert den Ablauf.

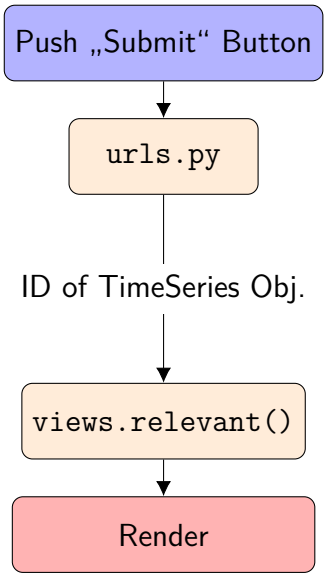

Abb. 3.13: Pogrammablauf: Manipulation der TimeSeries Objekte

Anfangs wird die URL im Skript `urls.py` zugeordnet und die passende Funktion ausgeführt. Bei Betätigen des „Submit“ Buttons werden die ID's von den drei „TimeSeries“ der Funktion übergeben und alle drei aus der zweiten Datenbank ausgelesen. Die Identifikation des zu manipulierenden Datenobjektes erfolgt über den Button. Jeder dieser Buttons hat einen eigenen Namen und kann die Objekten eindeutig identifizieren. Wenn das „TimeSeries“ Objekt geändert werden soll, wird der Eintrag „Relevance“ auf „1“ gesetzt. Bei Betätigung des „Decline“ Buttons wird durch die Funktion `irrelevant`, der Eintrag auf „0“ gesetzt und in der Datenbank abgespeichert.
Wie in Abbildung 3.3 dargestellt, erhalten die UserInnen Feedback in Form einer Meldung. Diese Meldung wird durch eine einfache `if/else` Bedingung im HTML-Code implementiert. Durch die Abfrage des „Relevance“ der „TimeSeries“ Objekte anhand ihrer ID kann überprüft werden, ob dieser noch auf dem Standardwert „2“ ist, oder schon bewertet wurde, also demnach „1“ oder „0“ beträgt. Der Standardwert „2“ dient dazu, die Buttons zur Klassifizierung des Plots sichtbar zu lassen.

4 Auswertung der Daten

Die zugrunde liegenden Zeitreihendaten stammen aus verschiedenen Sensoren, die Messwerte eines Bohrprozesses aufgenommen haben. Bei den UserInnen handelt es sich um MitarbeiterInnen der Technischen Universiät Wien, die sich im Zuge eines Projektes mit diesen Daten auseinandersetzen müssen und dadurch Wissen und Erfahrung mitbringen. Aus diesem Grund ist diese Gruppe ideal geeignet, um die Zeitreihendaten nach Relevanz zu klassifizieren. Jedem/Jeder Mitarbeiter/in wird genau eine Farbe zugeordnet. Durch die eindeutige Zuordnung der Farben grün, rot und blau trainiert jeder/jede Mitarbeiter/in den DecisionTree mit dem Featureset, das der jeweiligen Farbe zugeordnet ist. Es gibt keine Vorgabe, nach welchen Kriterien die Zeitreihen als relevant klassifiziert werden müssen. Die Einstufung, ob eine Zeitreihe als „relevant “ eingestuft wird, obliegt den UserInnen. Das Ziel ist es, von jedem „TimeSeries“ Typ 150 Objekte zu generieren.

Um die Auswirkungen der unterschiedlichen Featuresets auf den Machine Learning Algorithmus darzustellen, wird die Präzision berechnet. Mithilfe der Formel (2.1) kann die Genauigkeit bestimmt werden. Die Formel für die Genauigkeit lautet:

$$\epsilon = \frac{TP + TN}{TP + FP + FN + TN}$$

Die Daten für die Auswertung werden aus der zweiten SQLite Datenbank ausgelesen. Hierfür werden die Relevanzdaten der einzelnen „TimeSeries“ Objekte herangezogen. Um den Lernverlauf des Entscheidungsbaums grafisch darzustellen, wird die Genauigkeit nach jedem Datenobjekt berechnet. Die Auswertung ist in Abbildung 4.1 dargestellt.

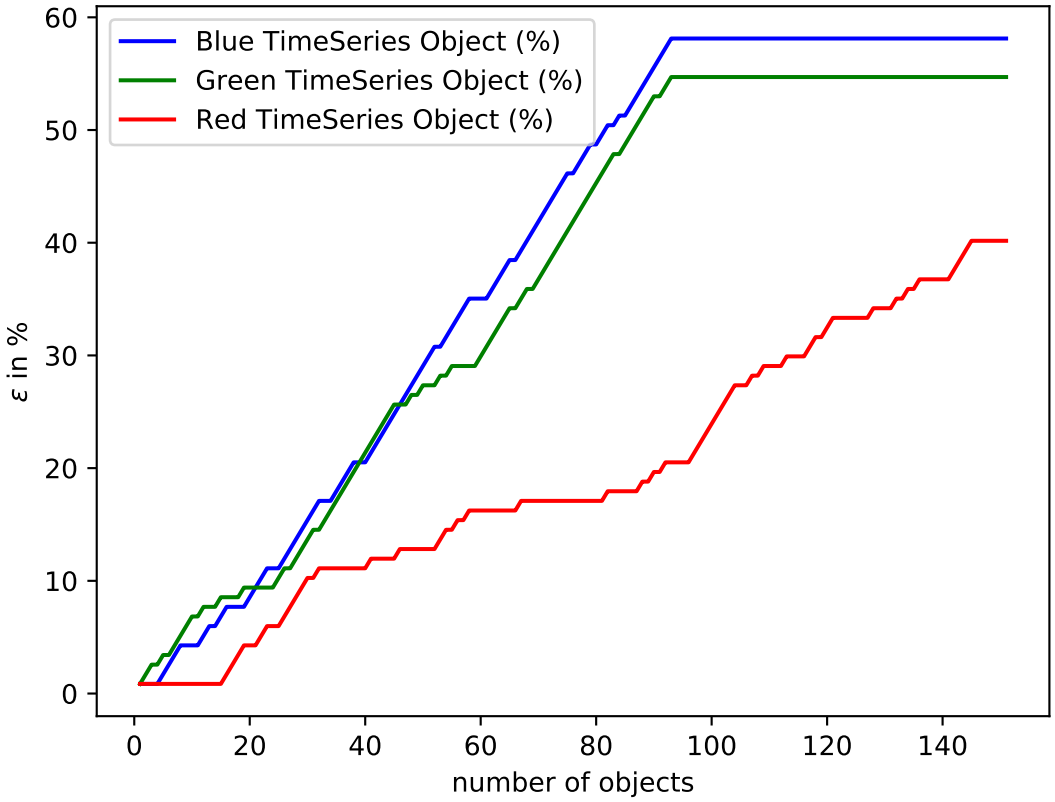

Abb. 4.1: Auswertung des Erfahrungszuwachs des Machine Learning Algorithmus

Aus der Grafik ist zu erkennen, dass die Genauigkeit der blauen „TimeSeries“ Objekte, die die Kombination der inhärenten und generierten Features beinhaltet, am höchsten ist. Die rote Kurve der Genauigkeitswerte ist flacher als jene des grünen und des blauen Objekte. In der Anfangsphase ist zu sehen, dass viele der grünen und blauen Plots als relevant bewertet werden, wohingegen die roten Plots erst ab Objekt 20 anfangen relevant zu werden. Dieser Effekt ist auf die Zufälligkeit der Datenwahl zurückzuführen, denn ohne Trainingsdaten kann der Machine Learning Algorithmus die Datenobjekte nicht klassifizieren und somit die dargestellten Plots nicht überprüfen.
Die blaue und grüne Kurve haben einen sehr ähnlichen Verlauf. Begründet werden kann diese Ähnlichkeit mit den gemeinsamen Features „Minimum“ und „Maximum“. Da auch die Verläufe fast durchgehend linear ansteigen, kann man davon ausgehen, dass der Lernprozess des Algorithmus durch die grünen und blauen Featuresets gut funktioniert. Die rote Kurve zeichnet sich durch eine flachere Steigung und einigen Flachpassagen aus.

Der Grund für die Flachpassagen sind entweder ein schlecht trainierter Algorithmus oder das Zufälligkeitsprinzip, das sehr häufig Datenreihen wählt, die als irrelevant eingestuft werden. Die Flachpassagen der grünen und blauen Kurve ab dem 90. Featureset sind auf technische Schwierigkeiten zurückzuführen. Die UserInnen, denen „Blau“ und „Grün“ zugewiesen wurde, hatten primär Probleme mit einer guten Serververbindung.

Neben dem Training des Machine Learning Algorithmus gibt es auch andere Faktoren, die das Ergebnis beeinflussen können. Einer dieser Faktoren ist die gewählte Methodik der Applikation. Durch die zufällig gewählten Zeitreihendaten werden immer unterschiedliche „TimeSeries“ Objekte generiert. Dadurch besteht die Möglichkeit, dass bei einer Farbe häufiger relevante Datenobjekte erzeugt werden, als bei den anderen und die UserInnen unterschiedlich relevante Plots angezeigt bekommen. Durch die unterschiedlichen Plots erhält der Machine Learning Algorithmus unterschiedliches Feedback von den UserInnen, das wiederum das Lernverhalten beeinflusst.
Ein weiterer Einflussfaktor ist die individuelle Einstufung der UserInnen. Die präsentierten Plots werden je nach Wissen, Erfahrung und persönliche Einschätzung unterschiedlich bewertet. Infolgedessen werden die „TimeSeries“ Objekte unterschiedlich relevant eingestuft und dem Algorithmus übergeben. Auf diese Weise wird das Lernverhalten zusätzlich beeinflusst, sodass möglicherweise ähnliche „TimeSeries“ Objekte unterschiedlich bewertet werden.

5 Fazit und Ausblick

Durch die verschiedenen Featuresets können unterschiedliche Lernverläufe des Machine Learning Algorithmus abgeleitet werden. Interessanterweise ist das Training durch die inhärenten Features (blau & grün) besser als durch die statistischen Features (rot).
Das hier erreichte Ergebnis muss jedoch kritisch betrachtet werden. Unter Umständen führen andere UserInnen und eine andere Methodik zu anderen Ergebnissen. Vor allem die zufällig gewählten Zeitreihendaten bei der Erstellung der Plots und der Erstellung der Datenobjekte sind ein nicht zu vernachlässigender Einflussfaktor, da dadurch das Training des Machine Learning Algorithmus mit unterschiedlichen „TimeSeries“ Objekten erfolgt.

Zudem ist die Methodik des Herausfilterns der relevanten Monitoringdaten ausschlaggebend für die Genauigkeit der Klassifizierung durch den Algorithmus. Die Methodik wurde jedoch bewusst unkompliziert gehalten, damit diese auch für andere Beteiligte, die mit der Materie „Machine Learning “ weniger vertraut sind, nachvollziehbar ist.

Durch das wachsende Interesse am Einsatz künstlicher Intelligenz in der Industrie gibt es weiterhin Bedarf, mit Monitoringdaten zu forschen. Neben des in dieser Arbeit vorgestellten Vergleiches verschiedener Featuresets anhand der Genauigkeit wäre es interessant, einen Vergleich anhand von Präzision, Trefferquote und Fehlerrate (Formeln (2.2) - (2.4)) zu machen, um die Klassifizierung des Machine Learning Algorithmus verifizieren zu können. Ein weiteres interessantes Forschungsgebiet ist der Einsatz von Deep Learning Algorithmen in der Klassifizierung von Monitoringdaten, um feststellen zu können, ob

dadurch auch eine sinnvolle Klassifizierung möglich ist. Ein weiterer großer Schritt wäre die Entwicklung einer Applikation, die in der Lage ist, mithilfe eines vorab trainierten Machine Learning Algorithmus, den UserInnen Plots auf Grundlage von gespeicherten Monitoringdaten anzeigen zu lassen. Damit könnten die gespeicherten Daten sinnvoll genutzt werden und vor allem Zeit und Kosten gespart werden.

Zukünftig wird die Verarbeitung von Daten mithilfe von künstlicher Intelligenz unumgänglich sein. Das Interesse an dem in der Datenmenge enthaltenen Wissen steigt stetig und es bedarf an guten Algorithmen, um dieses Wissen aus den Daten zu extrahieren. Zudem steigt die Menge an Daten, die täglich abgespeichert wird, rasant an, sodass es für Menschen fast unmöglich ist, die für sie relevanten Daten herauszufiltern. Deswegen ist umso wichtiger, weiter in diesem interdisziplinären Feld zu forschen, um den Umgang mit der immer größer werdenden Menge an Daten bewältigen zu können.

Aus der Sicht des Autors ergeben sich Fortführungen dieser Arbeit in zwei Richtungen. Einerseits wäre eine Ausweitung der Validierung der in dieser Arbeit vorgestellten Ergebnisse sinnvoll. Durch eine erhöhte Anzahl an „TimeSeries“-Objekten, durch andere Features oder durch die Wahl eines anderen Machine Learning Algorithmus können möglicherweise andere Lernverhalten beobachtet werden. Andererseits ist die Erstellung einer Applikation, die bereits einen trainierten Machine Learning Algorithmus im Hintergrund hat, sinnvoll, um relevante Zeitreihenabschnitte zu identifizieren. Die UserInnen validieren die Relevanz der Abschnitte.

Literaturverzeichnis

[1] Birgit Vogel-Heuser, Thomas Bauernhansl, and Michael Ten Hompel, editors. *Handbuch Industrie 4.0. Bd. 1: Produktion*. Springer Reference Technik. Springer Vieweg, Berlin, 2., erweiterte und bearbeitete auflage edition, 2017. OCLC: 951213977.

[2] M. Nero, C. Shan, L.-C. Wang, and N. Sumikawa. Discovering Interesting Plots in Production Yield Data Analytics. *ArXiv e-prints*, July 2018.

[3] Ankita Mangal and Nishant Kumar. Using big data to enhance the bosch production line performance: A kaggle challenge. *CoRR*, abs/1701.00705, 2017.

[4] Martin Hofmann, Florian Neukart, and Thomas Bäck. Artificial intelligence and data science in the automotive industry. *CoRR*, abs/1709.01989, 2017.

[5] Birgit Vogel-Heuser, Thomas Bauernhansl, and Michael Ten Hompel, editors. *Handbuch Industrie 4.0. Bd. 2: Automatisierung*. Springer Reference Technik. Springer Vieweg, Berlin, 2., erweiterte und bearbeitete auflage edition, 2017. OCLC: 958467762.

[6] Tatjana Lange and Karl Mosler. *Statistik kompakt*. Springer-Lehrbuch. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017.

[7] Bing Hu, Yanping Chen, and Eamonn Keogh. Time series classification under more realistic assumptions. In Joydeep Ghosh, Zoran Obradovic, Jennifer Dy, Zhi-Hua Zhou, Chandrika Kamath, and Srinivasan Parthasarathy, editors, *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 578–586. Society for Industrial and Applied Mathematics, Philadelphia, PA, May 2013.

[8] Eamonn J. Keogh and Michael J. Pazzani. Relevance feedback retrieval of time series data. pages 183–190. ACM Press, 1999.

[9] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *In proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*, pages 27–31, 1998.

[10] Alice Zheng. *Mastering Feature Engineering Principles and Techniques for Data Scientists (Early Release).* Oreilly & Associates Inc, 2017. OCLC: 957747646.

[11] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *SIGKDD'02*, pages 102–111, 2002.

[12] Ben D. Fulcher and Nick S. Jones. Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, December 2014.

[13] André Gensler and Bernhard Sick. Performing event detection in time series with SwiftEvent: an algorithm with supervised learning of detection criteria. *Pattern Analysis and Applications*, 21(2):543–562, May 2018.

[14] I. H. Witten. *Data mining: practical machine learning tools and techniques.* Elsevier, Amsterdam, fourth edition edition, 2017.

[15] Looking backwards, looking forwards: SAS, data mining, and machine learning. `https://blogs.sas.com/content/subconsciousmusings/2014/08/22/looking-backwards-looking-forwards-sas-data-mining-and-machine-learning/`. [Online; zugegriffen am 30. März 2018].

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] Biotrust | The difference between AI, Machine Learning and Deep Learning. `https://www.biotrustid.com/the-significant-difference-between-artificial-intelligence-machine-learning-and-deep-learning/`. [Online; zugegriffen am 17. August 2018].

[18] Dipanjan Sarkar, Raghav Bali, and Tushar Sharma. *Practical machine learning with Python: a problem-solver's guide to building real-world intelligent systems.* 2018.

OCLC: 1017098750.

[19] Oded Maimon and Lior Rokach, editors. *Data Mining and Knowledge Discovery Handbook*. Springer US, Boston, MA, 2010.

[20] Beniamino Murgante, Osvaldo Gervasi, Sanjay Misra, Nadia Nedjah, Ana Maria A. C. Rocha, David Taniar, Bernady O. Apduhan, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, and Gerhard Weikum, editors. *Computational Science and Its Applications – ICCSA 2012*, volume 7336 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[21] Esmael Bilal, Aranout Arghad, Fruhwirth Rudolf, and Thonhauser Gerhard. A Statistical Feature-Based Approach for Operations Recognition in Drilling Time Series. In *International Journal of Computer Information Systems and Industrial Management Applications. ISSN 2150-7988 Volume 5 (2013) pp. 454-461.*

[22] MacArthur, Brodley, and Chi-Ren Shyu. Relevance feedback decision trees in content-based image retrieval. pages 68–72. IEEE, 2000.

[23] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. pages 33–42. ACM Press, 1999.

[24] The Web framework for perfectionists with deadlines | Django. `https://www.djangoproject.com/`. [Online; zugegriffen am 18. April 2018].

[25] Sqlite home page. `https://www.sqlite.org/about.html`. [Online; zugegriffen am 13. Juni 2018].

[26] Jason Brownlee. Basis feature engineering with time series data in python. `https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/`, December 2016. [Online; zugegriffen 24. April 2018].

[27] Nikos E. Mastorakis and Stavros D. Nikolopoulos, editors. *Information processing and technology*. Nova Science Pub, Huntington, NY, 2001. OCLC: ocm49344368.

[28] Katja Specht, Rebecca Bulander, and Wolfgang Gohout. *Statistik für Wirtschaft und Technik*. Oldenbourg, München, 2012. OCLC: 845000583.

[29] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78, October 2012.

[30] Rosa L. Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H. Ngo. Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12(1):8, Feb 2012.

[31] Decisiontree classifier (sklearn). `http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.fit`. [Online; zugegriffen 13. Juli 2018].

Abbildungsverzeichnis

1.1 Grafische Darstellung der Problemstellung 3

2.1 Venn-Diagramm Data Mining, vgl. [15] 6

2.2 Entwicklung des Machine-Learnings, vgl. [17] 7

2.3 Schematische Darstellung eines Deep Learning Modells (Kategorisierung eines Bildes), vgl. [16] 9

2.4 Darstellung des ML-Modells 10

2.5 Übereinstimmung tatsächliche Klasse - Vorhersage 14

2.6 Feature Engineering im Machine Learning Prozess [10] 16

3.1 Struktur der Machine Learning Applikation 23

3.2 Screenshot des GUI: Fenster zur Klassifikation der Plots 25

3.3 Feedback nach Betätigen der Buttons 26

3.4 Screenshot des GUI: Auswahlfenster für unterschiedliche Featuresets . . . 27

3.5 Darstellung der statistischen Größe „Schiefe“ 29

3.6 Darstellung der statistischen Größe „Wölbung“ 30

3.7 Darstellung des Trainings (Bsp.: Featureset Blue) 34

3.8 Pogrammablauf: Erstellung der TimeSeries Objekte 36

3.9 Abfrage Anzahl der Trainingsdaten 37

3.10 Funktion: `handle_non_numerical_data` 38

3.11 Training des DecisionTree-Algorithmus 39

3.12 Überprüfungsmechanismus in der Funktion `get_data` 40

3.13 Pogrammablauf: Manipulation der TimeSeries Objekte 41

4.1 Auswertung des Erfahrungszuwachs des Machine Learning Algorithmus . 43

Tabellenverzeichnis

3.1 Darstellung des inhärenten Featuresets 28

3.2 Darstellung des generierten Featuresets 31

3.3 Darstellung des kombinierten Featuresets 32

A Anhang

- Hyperlink zum git-Repository:
  `https://bitbucket.org/zeit73/monitoring/src/master/`