technische universität
dortmund

Bachelorarbeit

Klassifikation von Zeitreihen über die Bestimmung häufiger symbolisierter Subsequenzen

Dustin Gärtner
5. Dezember 2014

Gutachter:
Prof. Dr. Katharina Morik
Dipl. Inform. Marco Stolpe

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS-8)
http://www-ai.cs.uni-dortmund.de/

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Hintergrund

In kaum einer anderen Branche sind Fehlproduktionen so kostspielig, wie in der Stahlindustrie. Selbst wenn fehlerhafte Ware wieder eingeschmolzen werden kann, sind andere Aufwände (Hilfsstoffe, Betriebsstoffe, Personalkosten etc.) verbraucht und haben dem entsprechend Kosten verursacht. Diese hohen Produktionsfehlkosten könnten gesenkt werden, falls fehlerhafte Ware schon vor der Fertigstellung aussortiert werden könnte. Da derzeit Qualitätskontrollen meist erst bei dem fertigen Produkt durchgeführt werden können und nicht schon während der Produktion, ist es nicht möglich, diese Kostensenkungen wahrzunehmen. Im Rahmen des Sonderforschungsbereichs 876, Projekt B3 existieren schon einige Arbeiten, welche die Korrelation von Produktionsverarbeitungsdaten und der Qualität des Endproduktes untersuchen. Es wird vermutet, dass dieser Zusammenhang existiert und anhand von Produktionsdaten eine Fehlererkennung möglich ist. Wenn dies der Fall ist, könnte eine Produktionsdatenanalyse während der Fertigung die fehlerhaften Produkte kennzeichnen und aussortieren. In einem darauf folgenden Schritt kann auch eine Überwachung des Prozesses erfolgen und eine Korrektur der Produktionsparameter automatisch vorgenommen werden. Diese präventiv Maßnahmen würden dann nicht nur zu einer Kostensenkung, sondern auch zu einer Gewinnsteigerung führen. Mit diesem Ziel, wurden die zur Verfügung gestellten realen Daten im Folgenden untersucht.

1.2 Zielsetzung

Die überlieferten Rohdaten sind als Zeitreihen gespeichert. Diese Reihen können den verschiedenen Sensortypen (Temperatur, Kraft, Drehzahl) zugeordnet werden, welche an unterschiedlichen Stellen des Produktionsprozesses montiert sind. Des Weiteren stehen zu den hier betrachteten Reihen die Qualitätseinstufungen des Endproduktes zur Verfügung. Der vielversprechende Rohdatensatz der Walzkraft, welcher an der Hauptwalze Daten aufzeich-

net, wird in dieser Arbeit genauer betrachtet (siehe Abbildung 2.1). Da die Betrachtung der kompletten Zeitreihen meist sehr aufwendig ist, werden die Daten in einem Vorprozess diskretisiert. Zudem wird durch die Diskretisierung das messbedingte Rauschen vermindert. Dabei sollen die Zeitreihen in handliche Fenster gestückelt werden, welche wiederum auf Symbole abgebildet werden. Dafür wird der in [9] vorgestellte Ansatz verwendet und mit einem Clustering-Ansatz für die Symbolisierung verglichen. Bei dem SAX Verfahren, werden die aus den Zeitreihen entnommenen gleich große Sequenzen, über die Mittelwertbildung in Symbole überführt. Bei dem anderen Verfahren wird der k-Means [3] Algorithmus angewendet, um zuvor generierte Fenster zu clustern (in Gruppen mit ähnlichen Eigenschaften einteilen). Dabei stellt jedes Cluster ein Symbol für das dieses Verfahren dar. Dieser zweite Ansatz verfolgt das Ziel, der Erhaltung der Forminformationen. Beispielsweise könnten drei Cluster entstehen, steigende, fallende und in Waage haltende Fensterreihenwerte. Dies ist der große Unterschied zu dem SAX Verfahren, welches nur die Mittelwerte vergleicht. Beide Verfahren haben den Vorteil der Dimensions-Reduktion, welches die Lernqualität steigert. An die Symbolisierung anschließend, werden die Häufigkeiten der einzelnen Symbole gezählt, woraufhin Vektoren mit gleicher Länge entstehen. Dabei wird auch eine n-Gramme Zählung vorgenommen. Bei einem Bigramm werden beispielsweise zwei aufeinanderfolgende Symbole als Wort gezählt. Dies wird mit überlappenden[1] sowie einer einfachen[2] n-Gramme Bildung untersucht. Diese Diskretisierung ist an der *Bag of Words* Repräsentation angelehnt. Durch diese Vorbehandlung, liegen die Daten nun in geeigneter Form vor, sodass die bekannten Lernmethoden (Naive Bayes, Decision Tree, k-NN), auf den Datensätzen schnell und effektiv angewandt werden können. Bei positiven Lernergebnissen können geeignete Cluster- und Symbolisierungsparameter extrahiert werden, welche wiederum die Produktion des deutschen Stahlherstellers optimieren kann.

1.3 Aufbau der Arbeit

In Kapitel 2 wird der Produktionsprozess des Stahlherstellers detailliert beschrieben. Die einzelnen Produktionsstationen und Sensoren werden vorgestellt und insbesondere die Blockwalze, an welcher der Datensatz dieser Untersuchung erhoben worden ist. Diese Daten werden vorgestellt, indem auf verschiedene Eigenschaften der Zeitreihen eingegangen wird. Des Weiteren wird auf bereits abgeschlossene Arbeiten des SFB 876 eingegangen, welche sich mit diesem o. ä. Datensätzen beschäftigen. In dem darauf Folgenden 3. Kapitel wird eine Einführung in die Zeitreihen Analyse gegeben. Hierbei wird sich auf die benötigten Definitionen beschränkt, für eine ausführliche Einführung wird [19] empfohlen. Zudem werden die benötigten Vergleichsmaße festgelegt und die später genutzten Lernverfahren aus den Data Mining Bereich vorgestellt. In Kapitel 4 werden die Vorverarbeitungsschritte detail-

[1] $\{A, B, C, D\} \rightarrow \{\{A, B\}, \{B, C\}, \{C, D\}\}$
[2] $\{A, B, C, D\} \rightarrow \{\{A, B\}, \{C, D\}\}$

liert beschrieben. Das SAX Verfahren und das k-Means Clustering wird dazu genutzt zuvor generierte Teilstücke der Datenreihen in Symbole zu überführen. Dabei werden einerseits Mittelwerte der Abschnitte genutzt und andererseits die zuvor generierte Clusterzentren. Kapitel 5 beschäftigt sich mit der Implementierung und Prozess Generierung in RapidMiner. Dort sind die einzelnen Prozesse für die Subsequenz Erzeugung, der Überführung in die Symbole sowie die anschließenden Lernverfahren präsentiert. Das 6. Kapitel beschäftigt sich mit den Experimenten auf den Rohdaten. Beginnend mit der Vorstellung der Parameter Kombinationen, der Lerndatengenerierung, wird dieses Kapitel die darauf erhobenen Ergebnisse mit einem schwer zu übertreffenden Default Lerner vergleichen. Abschließend mit Kapitel 7 wird die Arbeit ein Fazit zu den unterschiedlichen Symbolisierungsarten geben und einen Ausblick für weiterführende Studien zeigen.

Kapitel 2

Datenerhebung

Die Daten eines Stahlherstellers wurden an verschiedenen Stellen des Produktionsprozesses erhoben. Der Fertigungsprozess ist vereinfacht in Abbildung 2.1 dargestellt und wird im ersten Abschnitt von diesem Kapitel erläutert. Nachdem die Fertigung mit samt der vorhandenen Sensoren erläutert worden ist, werden in Kapitel 2.2 die zu untersuchenden Daten beschrieben. Es beginnt mit dem groben Verlauf und den Eigenschaften der Zeitreihen und endet mit der Antwort auf die Frage, warum ein direkter Vergleich nicht möglich ist.

2.1 Produktionsprozess

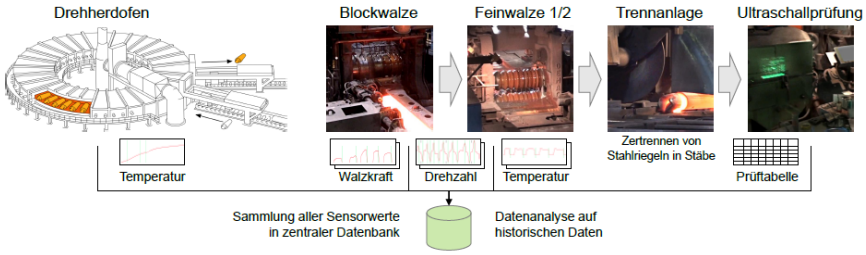

Abbildung 2.1: Produktionsprozess (Grafik: Marco Stolpe)

An der ersten Stelle der Produktion ist der Drehherdofen. Dieser besitzt üblicherweise 20 bis 30 Kammern. An der ersten Position wird ein Stahlrohling (hier in Form eines Blockes) in den Ofen gerollt. Dieser Rohling wird durch den Drehherdofen befördert. Dabei steigt die Temperatur im Inneren der jeweiligen Kammer an, um den Block zu erhitzen. Es existieren fünf verschiedene Temperaturzonen, welche jeweils mit einem Lufttemperatursensor ausgestattet sind. Diese Sensoren zeichnen innerhalb der Kammern stetig die Temperatur auf, welche durch ein mathematisches Modell auf die Temperatur des Blockinneren geschätzt werden kann. So steigt die Temperatur pro Zone weiter an, bis der Block eine Temperatur von rund 840° bis 880° Celsius erreicht hat. Nachdem der nun rot glü-

hende Block auf Warmwalztemperatur gebracht worden ist, verlässt dieser den Drehherdofen an letzter Position und wird zur Block- oder auch Warmwalze gerollt. Diese walzt den Block meist in eine quadratische Form, sodass der Block dadurch das gewünschte Profilmaß (Breite und Höhe) erreicht. Dies wird durch das Reversierverfahren der Blockwalze geleistet. Dabei wird zunächst der Block mit einer hohen Anstellung der Walze, vorwärts durch das Walzgerät gepresst. Dies wird im Folgenden als ein Walzstich bezeichnet. Anschließend wird die Walzanstellung geändert, sodass der Abstand zwischen den beiden Rollen verringert wird. Darauf wird der Block rückwärts wieder durch die Walze gerollt. So wird in mehreren Schritten (vor und zurück, vor und zurück...) die Vorform des Blocks gewalzt, wobei ein Walzstich maximal nur wenige Sekunden dauert. Dabei sind die einzelnen Rollen wassergekühlt. Da trotz ihrer speziellen Härtung und Verarbeitung, die heißen Stahlblöcke die Blockwalze schnell beschädigen würden. An der Blockwalze werden von mehreren Sensoren Daten aufgezeichnet, dabei handelt es sich um einen Temperatur-Sensor, der die Temperatur außerhalb (möglichst nah) des Stahls misst. Des Weiteren steht die Walzkraft sowie die Drehzahl der Blockwalze zur Verfügung. Wenn nun die gewünschte Vorform erreicht ist, wird der Stahlblock weiter zur Feinwalze gerollt. Dabei kann eine erneute Erwärmung des Stahls in einem Ofen von nöten sein. Falls die benötigte Temperatur für die Feinwalze nicht mehr vorhanden ist, wird dies von einem Aufseher initiiert (nicht in der Abbildung zu sehen). Wenn nun der Stahlblock die Feinwalze erreicht hat, wird auch wieder im Reversierverfahren der Block auf das gewünschte Profil (beispielsweise U oder L) gewalzt. Auch hier wird kleinschrittig die Höhe Walzanstellung verringert und dieselben Daten, wie an der Blockwalze erhoben. Anschließen wird in einer Trennanlage, der profilierte Stahlblock in genormte Längen geschnitten, diese Teilstücke werden Stäbe genannt. Die Stäbe werden im letzten Produktionsschritt mit einer Ultraschallprüfung auf Rissfreiheit und Lunker überprüft. Lunker ist ein Überbegriff für Verunreinigungen durch Schlacke oder Lufteinschlüsse. Anhand dieser Prüfung werden die Produkte bewertet und es können Stäbe oder Chargen aussortiert werden, wenn diese den gestellten Anforderungen nicht genügen.

2.2 Darstellung der Sensordatenwerte

In dem beschriebenen Prozess ist zu erkennen, dass einige autonome Stationen der Fertigung existieren. So können auch an verschiedenen Stellen unterschiedliche Fehler auftreten, sei es, während des Walzvorgangs in der Blockwalze, oder bei einem der beiden Feinwalzstationen. Eine nicht korrekte Erwärmung in einem der beiden möglichen Öfen, könnte auch ein fehlerhaftes Produkt zur Folge haben. Jedoch wurden für die weitere Betrachtung, nur die Walzkraftmuster der Blockwalze untersucht. Dies hat mehrere Gründe: Einerseits wurde von [8] bestätigt, dass beispielsweise der Drehzahlsensor direkt von der Walzkraft abhängt und damit keine neuen Erkenntnisse liefert. Andererseits steigt die

Komplexität der Versuche mit jedem weiteren betrachteten Sensor. Eine ausführliche Darstellung der Probleme beim verteilten Lernen in einem Produktionsprozess befindet sich in [1]. Die unterschiedlichen Symbolhäufigkeiten aller Sensoren müssten in einen Vektor gebracht werden, um die bekannten Lernverfahren zu nutzen. Die in dieser Arbeit betrachteten Optionen: $|Fensterzahl_B| \times |Symbolanzahl_B| \times |Symbolisierungsverfahren_B| \times |Lernverfahren_B| \times |n-Gramme_B| \times |überlappen_B|$, stehen nur für die Blockwalze entnommenen Daten. Für jede weitere Station / Sensor würden diese Kombinationen dazu kommen. Bei nur einer booleschen Option wären es gleich doppelt so viele Möglichkeiten. Dieser Mehraufwand würde den Zeitrahmen dieser Arbeit sprengen. Durch diese Sensoreinschränkung bleiben natürliche Abhängigkeiten zwischen den Stationen und Sensoren unberücksichtigt, dies soll aber als Hinweis auf eine längere Studie betrachtet werden.
Ein Musterdatensatz der Walzkraft ist in Abbildung 2.2 abgebildet. Während der Datenerhebung misst der Sensor die Kraft, mit der die Walze auf den Block einwirkt. Die Höchstwerte (*peaks*) sind die realen Einwirkungen auf den Stahl, die Schwankungen um den Nullpunkt zeigen an, dass keine Einwirkung auf das Blech besteht. Diese Fluktuationen können von dem Umstellen der Walze, dem vibrieren oder ähnlichen kommen. Die einzelnen peaks sind die Walzstiche, welche später auch herausgefiltert werden. Hier lässt sich auch schnell erkennen, warum ein direkter Vergleich nicht möglich ist. Die Walzstiche starten zu verschiedenen Zeitpunkten t_i bei divergenten Produktionen. Auch bei gleichen Startpunkten der Fertigung, ist der direkte Vergleich nicht möglich, denn jedes Walzstichintervall beinhaltet ein variable Anzahl an Werten. Sogar die Schritte des Reversierverfahrens (Walzstiche) und der Walzkraftintensität je Zeitreihe sind unterschiedlich. Von diesen

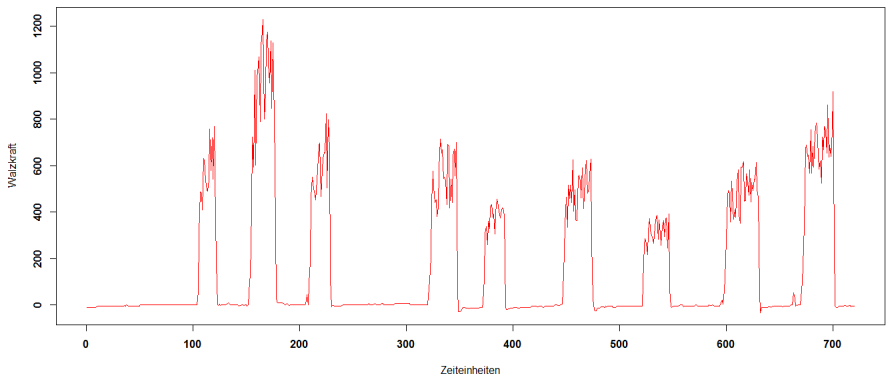

Abbildung 2.2: Nachgezeichnetes Walzkraftmuster einer tadellosen Produktion

Walzkraftmustern wurden etwa 400 korrekte Zeitreihen übertragen sowie dessen Endeinstufung der Qualität aus der Ultraschallprüfung. Es gab nur wenig fehlerhafte Riegel(etwa $\frac{1}{13}$), was auf den ersten Blick positiv erscheint. Wenn zwei Zeitreihen gegenüber gestellt werden, eine positiv und eine negativ bewertete, ist kein gravierender Unterschied zu se-

hen (vgl. Abbildung 2.3). Daraus lässt sich schließen, dass minimale Abweichungen die Qualitätsunterschiede ausmachen. Da nur eine kleine Menge wirklich fehlerhafter Produkte hergestellt wurden, liegt weniger Vergleichsmaterial vor, welches die Filterung auf die fehlerhaften Segmente erschwert. Jedoch weist [5] in seiner Arbeit darauf hin, das SAX

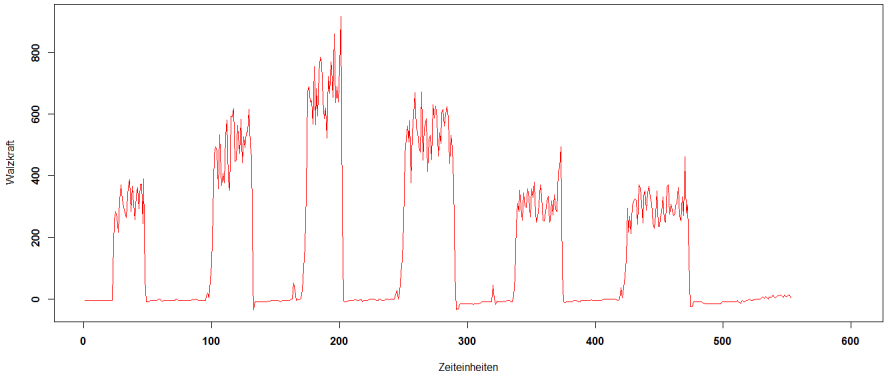

Abbildung 2.3: Nachgezeichnetes Walzkraftmuster einer Fehlproduktion

auch noch sehr gut auf diesen schmal besetzten Datensätzen arbeiten kann. Des Weiteren wird gerade in diesem Artikel eine Möglichkeit betrachtet *discords* (Unstimmigkeiten) in den zu untersuchenden Zeitreihen zu finden.

Kapitel 3

Einführung in die Zeitreihen Analyse

In diesem Kapitel werden die benötigten Grundlagen der Zeitreihen-Analyse erläutert. Nach den grundsätzlichen Definitionen werden benötigte Schwerpunkte ausformuliert. Sei es die Z-Normalisierung oder das *Sliding Window*. Anschließen werden die beiden Distanzmaße, die Euklidische Distanz sowie das Dynamic Time Warping, erläutert. Zum Ende des Kapitels werden die genutzten Lernverfahren vorgestellt.

3.1 Grundlagen der Zeitreihenanalyse

Die vorhandenen Daten sind in Form einer besonderen Wertereihe gegeben, der Zeitreihe.

3.1.1 Definition ***(Wertereihe)*** *Unter einer Wertereihe oder Reihe versteht man eine Abbildung*

$$x : \mathbb{N} \to \mathbb{R} \times \mathbb{C}^m$$

Anstelle von x(n) schreiben wir x_n und für eine endliche Folge der Länge n schreiben wir $(x_i)_{i\in\{1,...,n\}}$.

Diese Definition aus [13] ist sehr allgemein gehalten, sodass diese Definition weiter spezialisiert werden kann (wie beispielhaft in [15]).

3.1.2 Definition ***Zeitreihe*** *Unter einer Zeitreihe t wird eine geordnete reellwertige Liste $t = [t_1, t_2, ..., t_n], t_i \in R$ beliebiger Länge n verstanden. Diese werden häufig als Graph der Funktion $i \to t_i, i \in [1, ..., n]$ visualisiert.*

Die zu untersuchenden Wertereihen liegen in der univariaten Form vor, das bedeutet pro Element der Zeitreihe existiert nur ein Datenwert, hier in Form des aufgezeichneten Sensorwertes. Die Vorgegebenen Rohdaten liegen in einer multivariaten Zeitreihe, da pro Zeitpunkt mehrere Sensorwerte erfasst worden sind. Da die Messpunkte in einem kontinuierlichen Zeitabstand genommen worden sind, haben die Indizes der Zeitreihen die Angabe

der Zeit inhärent. Eine zeitliche unstetige Messung könnte durch eine Zeitdimension in den einzelnen Vektoren $\vec{t_i}$ gekennzeichnet werden. Wie in den vorliegenden Daten zu sehen ist, können die Zeitreihen sehr unterschiedliche Ausprägungen haben, solche zu vergleichen ist sehr kompliziert. Laut [4] ist es sogar bedeutungslos nicht normalisierte Zeitreihen zu vergleichen. Die dort angesprochene Normalisierung ist als Z-Normalisierung bekannt.

3.1.3 Definition ***z-normalisiert***[1] *Gegeben eine Zeitreihe t. t ist z-normalisiert wenn*

$$\mu := \frac{1}{n}\sum_{i=1}^{n} t_i = 0 \ , \ \sigma^2 := \frac{1}{n}\sum_{i=1}^{n}(t_i - \mu)^2 = 1$$

gilt.

Jede Zeitreihe kann auf eine z-normalisierte Zeitreihe abgebildet werden.

$$t'_i = \frac{t_i - \mu}{\sigma} | 1 \leq i \leq n$$

Für die später erläuterten Symbolisierungsverfahren werden keine ganzen z-normalisierten Zeitreihen benötigt, sondern nur Teilstücke dieser. Dafür sind in [5] zwei weitere anschauliche Definitionen gegeben. Einmal die Subsequenz, welche ein Stück aus einer Zeitreihe beschreibt und das *Sliding Window*, welches alle Subsequenzen aus einer Zeitreihe definiert.

3.1.4 Definition ***Subsequenz*** *Gegeben sei eine Zeitreihe T der Länge m. Eine Subsequenz C von T ist ein Teilstück der Länge $n \leq m$, beginnend von einer Position p, sodass gilt: $C = t_p, ...t_{p+n-1}$ für $1 \leq p \leq m - n + 1$.*

Eine so definierte Subsequenz ist wiederum eine Zeitreihe. Es wird jedoch von nöten sein alle möglichen Subsequenzen aus den Zeitreihen zu entnehmen.

3.1.5 Definition ***Sliding Window*** *Gegeben sei eine Zeitreihe T der Länge m und eine Nutzerdefinierte Subsequenz der Länge n. Alle möglichen Subsequenzen die aus T mithilfe eines Sliding Window entnommen werden können bilden die Menge C_q.*

Die so extrahierten Subsequenzen können nun in dem folgenden Schritt verglichen werden.

3.2 Distanzmaße

Der Vergleich divergenter Zeitreihen, erfordert ein Ähnlichkeitsmaß. Dieses Maß kann in Form des Abstands definiert werden. Für den weiteren Verlauf dieser Arbeit werden insbesondere zwei Abstandsmaße genutzt. Zunächst wird die Euklidische Distanz vorgestellt. Diese vergleicht je zwei Elemente (auch Attribute) miteinander, indem es den Abstand im Raum misst. Hierfür werden Zeitreihen gleicher Länge benötigt, wobei ein einfaches

[1] aus [15]

Auffüllen der zu kurz geratenen Reihen mit unterschiedlichen Techniken denkbar wäre. Im Anschluss daran wird das Abstandsmaß Dynamic Time Warping ausgeführt. Dieses lässt ungleich lange Reihen zum vergleichen zu. Bei heterogenen Eingaben wird dann lediglich ein Attribut mit mehreren anderen verglichen. Bei homogener Eingabe unterscheidet sich das Maß trotzdem von der Euklidischen Distanz, da das Dynamic Time Warping versucht ein möglichst gutes Paar zu finden. So kann die zweite Methode Zeitreihen auf ihre Ähnlichkeit untersuchen und nicht nur auf den räumliche Abstand (siehe Abbildung 3.1).

3.2.1 Euklidische Distanz

Bei der euklidischen Distanz werden die einzelnen Merkmale aus dem Merkmalsraum jeweils verglichen. Die Differenz wird quadriert und aufsummiert. Aus der Summe wird die Wurzel gezogen, dies ist dann der Abstand d_E zwischen $\vec{x}$ und $\vec{y}$. Hierfür müssen natürlich alle Vektoren numerisch sein und gleicher Länge oder zuvor auf ein vergleichbares Maß gebracht werden.

3.2.1 Definition *Die Euklidische Distanz ist definiert als:*

$$d_E(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)}$$

Die Einschränkung auf Zeitreihen gleicher Länge kann in der Realität sehr problematisch werden. Die gegebenen Daten sind beispielsweise nicht von gleicher Länge, das bedeutet, eine Anwendung der Euklidischen Distanz ist zunächst nicht möglich. Es existieren mehrere Wege diese trotz unterschiedlicher Längen möglich zu machen. Die größere Zeitreihe könnte beschnitten werden, jedoch führt dies zum Verlust von Daten. Die kürzere Zeitreihe kann ebenso gut erweitert werden, um die gewünschte Länge zu erreichen. Die Datengenerierung kann auf unterschiedliche Art und Weise geschehen: auffüllen mit dem Mittelwert der gesamt Daten, dem Minimum oder Maximum, schreiben von Nullen oder einem fixen Wert. Jeder dieser Möglichkeiten führt zur Verfälschung der Daten, kann aber nach Sichtung und Interpretation dieser zu einem guten Ergebnis führen. Beispielsweise sind die Vorliegenden Daten der Blockwalze meist um den Nullpunkt, nur bei den einzelnen Walzstichen wird von diesem im großen Maße abgewichen. So würde die vermutlich sinnvollste Methode, das auffüllen mit nullen sein. Da zu diesen Zeitpunkten keine Walzkraft auf den Block einwirkt, wäre dies am realistischsten. Jedoch verzerrt dies sehr das Distanzergebnis, falls die Zeitreihen eine unterschiedliche Anzahl der Walzstiche haben. Da $d_E(\vec{0}, \vec{t})$ bei vorhanden Walzstichen sehr groß werden würde, könnte man annehmen es wäre sinnvollen den vorherigen Mittelwert der Zeitreihe zu wählen. Dies würde am geringsten das eigentliche Distanzmaß verändern. Diese ganze Problematik der ungleichen Zeitreihen wird meistens durch die Einschränkung auf gleich lange Zeitreihen umgangen. Um mit Zeitreihen ungleicher Länge zu arbeiten, wird das Distanzmaß Dynamic Time Warping vorgestellt. Welches ohne das hinzufügen oder bescheiden der zugrunde liegenden Daten auskommt und Ähnlichkeit zwischen diesen Zeitreihen sucht.

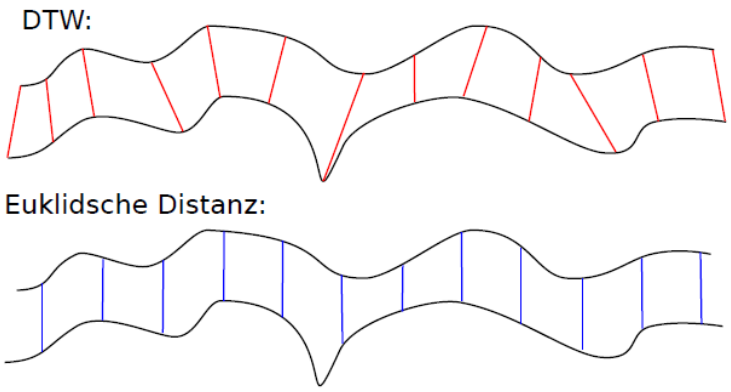

Abbildung 3.1: Vergleich DTW Distanz zu Euklidischen Distanz [17]

3.2.2 Dynamic Time Warping

Das Distanzmaß Dynamic Time Warping, kurz DTW wurde in den 1970iger Jahren entwickelt. Ursprünglich wurde das Verfahren zur Spracherkennung genutzt [18]. Dafür wurde eine korrekte Erkennung der Wörter, mit unterschiedlich lang gesprochenen Buchstaben, benötigt. Obwohl die Zeitreihen unterschiedlicher Länge sind, findet das DTW immer noch Ähnlichkeiten zwischen diesen. Durch die Bestimmung des optimalen *Alignment* (siehe 3.1) zu jedem Element in der Zeitreihe, benötigt das Verfahren jedoch lange Laufzeiten. Im Folgenden wird zunächst das Klassische DTW aus [14] erläutert und anschließend einige Laufzeit Verbesserungen vorgestellt.

Das Klassische DTW sucht das Minimum aus den Warping-Pfaden. Diese Pfade sind die zuvor angedeuteten Alignments zwischen den Zeitreihen.

3.2.2 Definition ***Warping-Pfad*** *Gegeben sind zwei Zeitreihen t, s der Länge n, m. Der Warping-Pfad $p_{t,s}$ ist eine Liste von Indexpaaren $p = (p_1, ..., p_L$ mit $p_i = (n_i, m_i) \in [1 : n] \times [1 : m]$ für $i \in [1 : L]$, sodass gilt:*

- *Grenzbedingung:* $p_1 = (1, 1), p_i = (n, m)$
- *Monotonie:* $n_1 \leq n_2 \leq ... \leq n_L, m_1 \leq m_2 \leq ... \leq m_L$
- *Schrittgröße:* $p_{l+1} - p_l \in (1, 0), (0, 1), (1, 1)$ *für* $l \in [1 : L - 1]$

Die Gesamtkosten eines Warping-Pfades betragen:

$$cost(p_{t,s}) = \sum_{(i,j)\in p_{t,s}} (t_i - s_j)^2$$

Von den errechneten Warping Pfaden sucht nun das Dynamic Time Warping den kostengünstigsten Pfad $p_{t,s}$.

3.2.3 Definition ***Dynamic Time Warping*** *Seien t, s Zeitreihen der Länge $n, m > 0$. Dann ist:*

$$d_{DTW}(t,s) = min\{cost(p_i)|p_i \in p_{t,s}\}$$

Um einen minimalen Warping Pfad zu berechnen, wird eine Laufzeit von $O(n+m)$ benötigt. Jedes Element aus t muss mit jedem aus s verglichen werden. Diese Quadratische Laufzeit ist insbesondere mit der Folgenden Anwendung, des k-Means Clustering inakzeptabel, da bei dem Clustering viele dieser DTW Distanzen berechnet werden müssen. Laut [16] ist gerade das Anwenden des DTW der Flaschenhals der meisten Verfahren. Aufgrund dessen sind in [15] einige Laufzeit Verbesserungen zusammen getragen und angewandt worden. Zunächst werden dort Grenzen der Warping-Pfade diskutiert. Hierfür wird das *Sakoe-Chiba-Band* [18] als auch das *Itakura-Parallelogramm* vorgestellt, da diese den Suchraum der möglichen Warping-Pfade einschränken. Darauf folgen die Betrachtung von unteren Schranken von Kim [7], sowie Keogh[6], welche ebenfalls eine Laufzeit Verbesserung des DTW anstreben.

All diese angesprochenen Effizienzsteigerungen wurden in [15] ausführlich beschrieben und getestet. Letztendlich wurden die Ergebnisse in ein RapidMiner Plug-in integriert, welches ich bei meinen späteren Experimenten nutze.

3.3 Überwachtes Lernen

Das überwachte Lernen ist ein Teilgebiet des Data Mining. Dabei benötigen die Lernverfahren ein zuvor richtig klassifizierten Testdatensatz, anhand dessen diese die Stichprobe lernen kann. Anschließend wird anhand der errechneten Regeln, der Gesamtdatensatz klassifiziert. Im letzten Abschnitt dieses Kapitels werden kurz die beiden genutzten Klassifizierer vorgestellt. Dabei wird für eine erschöpfende Einführung auf [22] verwiesen.

3.3.1 Naive Bayes Algorithmus

Der Naive Bayes Klassifikator, ist wegen der schnellen und einfachen Berechnung und meist doch zuverlässigen Klassifizierung, ein beliebtes Lernverfahren. Es ist nach dem zugrundeliegenden Theorem von Bayes benannt. Bei diesem Lernverfahren werden alle Objekte O der Klasse C zugeordnet, zu welcher sie am Wahrscheinlichsten gehören. Diese Methode setzt zwar voraus, dass alle Merkmale voneinander unabhängig sind, was in der Realität sehr selten ist, aber es erzielt trotzdem gute Ergebnisse [14].

Dass ein Objekt in der Klasse C liegt, auf die einzelnen Attribute (m_i) bezogen, lässt es sich durch das Theorem von Bayes darstellen:

$$P(C|m_1...m_n) = \frac{P(C) * P(m_1...m_n|C)}{\sum_{c \in C} P(C=c)P(m_1...m_n|C=c)} \tag{3.1}$$

3.3.2 k-nächste-Nachbarn

Bei dem k-NN Algorithmus, werden die Merkmalsvektoren mit den k nächsten, bereits zugeteilten Vektoren verglichen. Der zu überprüfende Vektor $\vec{x}$ wird der Klasse C zugeordnet, zu welcher die meisten unmittelbaren Nachbarn von $\vec{x}$ gehören. Zunächst werden die im Korpus enthaltenen Beispiele korrekt zugeordnet und in den $n - Dimensionalen$ Raum abgebildet. Anschließend werden einzeln, die zu klassifizierenden Objekte $\vec{y_i}$ in dem Raum platziert und mit den k nächsten Nachbarn verglichen. Hierfür wird als Distanzmaß meist die Euklidische Distanz verwendet. Das neue Objekt wird der Klasse zugeordnet, zu welcher die meisten seiner Nachbarn gehören. Folgerichtig ist das bei einer Binären Klassifizierung ein ungerades k zu wählen ist, sodass kein Objekt gleich viele Nachbarn von C_0 und von C_1 hat. Des Weiteren sollte in der Trainingsmenge (Korpus) alle Klassen vertreten sein.

Kapitel 4

Symbolisierungsverfahren

In Kapitel 4 dieser Arbeit werden die zwei angewendeten Symbolisierungsverfahren vorgestellt. Es beginnt mit dem **S**ymbolc **A**ggregate Appro**X**iomation, welches die Zeitreihen in gleichgroße Fenster teilt, dessen Mittelwerte bildet und diese als Grundlage der Symbolisierung wählt. Abschließend wird das k-Means Clustering vorgestellt. Dieses unüberwachte Lernverfahren teilt die zu untersuchende Menge in k Cluster ein, wobei der Abstand der innewohnenden Clusterelemente zueinander minimiert wird. Die darauf Folgende Symbolisierung der Clusterelemente soll mit dem SAX Verfahren in dem Experimentalteil verglichen werden. Zuletzt werden noch verschiedene Darstellungen der Symbolhäufigkeiten vorgestellt.

4.1 SAX - Symbolic Aggregate ApproXimation

Das **S**ymbolc **A**ggregate Appro**X**iomation, kurz SAX verfahren wurde 2003 von Lin et. al. vorgestellt [9] und seitdem weiterentwickelt. Es gibt viele weitere Arbeiten [5] [10] [12] [20] unter anderem von Eamonn Keogh, welche sich mit diesem Symbolisierungsverfahren beschäftigen. Beispielsweise untersucht [5], aus dem Jahre 2005, Zeitreihen nach ihrer ungewöhnlichsten Subsequenz. Dabei wird zunächst ein *Brute Force* Ansatz vorgestellt, welcher eine Laufzeit von $O(m^2)$ hat. Der Algorithmus wird mit dem geordneten Ergebnis des SAX aufgerufen, sodass dieser weitaus schneller abschneidet. Die vorliegenden Daten könnten ebenfalls auf *discords* untersucht werden, wenn diese Versuche zu keinem Ergebnis führen. Das Symbolisierungsverfahren arbeitet in zwei Schritten. Für jedes Fenster wird der Mittelwert über die gegeben Werte gebildet und anschließend wird jedem dieser Fenstermittelwerte ein Symbol zugeordnet. Dabei wird davon ausgegangen, dass eine Normalverteilung der Werte vorliegt. Falls diese Zeitreihendaten nicht gaussverteilt sind, arbeitet der Algorithmus trotzdem korrekt, es schmälert somit die Effizienz von SAX. Diese strikte Art der Zuweisung kann jedoch von Nachteil sein. Nicht immer sind die gegebenen Daten normalverteilt, sodass SAX vermutlich kaum gute Ergebnisse liefert. Die zweite hier

vorgestellte Methode liefert dort einen flexibleren Ansatz, welches keine harten Grenzen an die Symbolverteilung stellt. Bei beiden Verfahren müssen die Daten außerdem vorher z-normalisiert werden, dadurch ist der Mittelwert Null und die Standardabweichung Eins. Dies wird hier insbesondere bei der Symbolzuweisung benötigt, da dort mit statischen *breakpoints* gearbeitet wird. Außerdem können so unterschiedliche Zeitreihen(-Segmente) besser auf ihre Form verglichen werden.

4.1.1 Piecewise Aggregate Approximation

Die normalisierte Zeitreihe wird im ersten Schritt, der Mittelwertbildung über die markierten Fenster, auf eine Länge von $\frac{n}{w}$ abgebildet, wobei n die Länge der Zeitreihe darstellt. Dabei bekommt der Algorithmus nur den Parameter w übergeben, welcher die Dimension des Ergebnisses darstellt. Es gilt für jedes Fenster:

$$window_i = (t_{i-1*\frac{n}{w}+1}, ..., t_{i*\frac{n}{w}})$$

Die einzelnen Fenster können nun durch die Länge der Subsequenz dividiert werden. Dieser Schritt wird Piecewise Aggregate Approximation, kurz PAA genannt (siehe Abbildung 4.1).

$$\overline{t_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j \tag{4.1}$$

Die Zeitreihe T kann in einem w-Dimensionalen Vektor dargestellt werden. Dabei wird jedes Element $\bar{t}_i$ wie in 4.1 berechnet. Die transformierte Zeitreihe $\overline{T} = \bar{t}_1, ..., \bar{t}_w$ kann nun im zweiten Schritt des SAX weiter verarbeitet werden.

4.1.2 Diskretisierung

Die Annahme das die Zeitreihendatenwerte gaussverteilt sind [9], wird sich nun zunutze gemacht mit der Aussage, dass nach diesem Schritt jede Symbolausprägung α etwa gleich häufig im Ergebniswort vorhanden ist. Dieser Schritt wird *Discretization* genannt und beginnt mit der Aufteilung des Wertebereichs in a Intervalle. Hierfür wird aus einer Tabelle die einzelnen *breakpoints* extrahiert und gesetzt (siehe Abbildung 4.2).

4.1.1 Definition *Breakpoints* *Die Breakpoints sind in einer geordnete Liste von* $B = \beta_1, ..., \beta_{a-1}$ *angegeben, sodass unter einer* $N(0,1)$ *Gaußverteilung je* $\frac{1}{a}$ *Elemente in dem Intervall* β_i *bis* $\beta_i + 1$ *liegen. Dabei sei* $\beta_0 = -\infty$ *und* $\beta_a = \infty$.

Anschließend kann jedem Mittelwert aus $\overline{T}$ einem Symbol zugeordnet werden. Dabei wird $\bar{t}_i$ welches zwischen β_l und β_{l+1} liegt, auf das Symbol α_{l+1} abgebildet. Das resultierende Wort $\{\alpha_i\}^*$ ist die Symbolisierte Zeitreihe $\overline{T}$.

4.1.2 Definition *Wort* *Gegeben sei ein Subsequenz* T *der Länge* n. *Diese kann auf ein Wort* $\hat{T} = \hat{t_1}, ..., \hat{t_w}$ *wie Folgt abgebildet werden. Sei* $alpha_j$ *das* j. *Element eines Alphabetes. So kann nach der PAA Transformierung* $\overline{T}$ *zu* $\hat{T}$ *überführt werden:*

$$\hat{t_i} = \alpha_i \leftrightarrow \beta_{j-1} \leq \overline{t_i} < \beta_j$$

Zur Veranschaulichung wurden folgende Grafiken aus [10] übernommen. In der Abbildung 4.1 ist der erste Schritt, das Piecewise Aggregate Approximation dargestellt. In der Abbildung ist die zu Symbolisierende Zeitreihe C sowie dessen gemittelte Zeitreihe $\overline{C}$ dargestellt (oben). Es wurden 8 Subsequenzen (unten) generiert $\overline{c_i}$, welche jeweils $\frac{1}{8}$ der Zeitreihendaten beinhalten. Anhand dieser Subsequenzen wurden aus den 128 Datenwerten die Mittelwerte gebildet, welche in der roten Stufenfunktion im Graphen eingetragen worden ist. Der zweite Schritt wird in der Abbildung 4.2 präsentiert. Die beiden breakpoints teilen den Wertebereich in 3 gleich gut besetzte Intervalle ein. Diese Intervalle wurden mithilfe der Gaussglocke bestimmt. So wird gewährleistet, dass bei einer normalverteilten Eingabe die annähernd gleiche Anzahl Symbole generiert werden. Wie in der Abbildung zu sehen, werden die Mittelwerte jeweils den Intervallen, welche die breakpoints einschließen, zugewiesen. Es wird das Wort *baabccbc* aus der untersuchten Zeitreihe C erstellt.

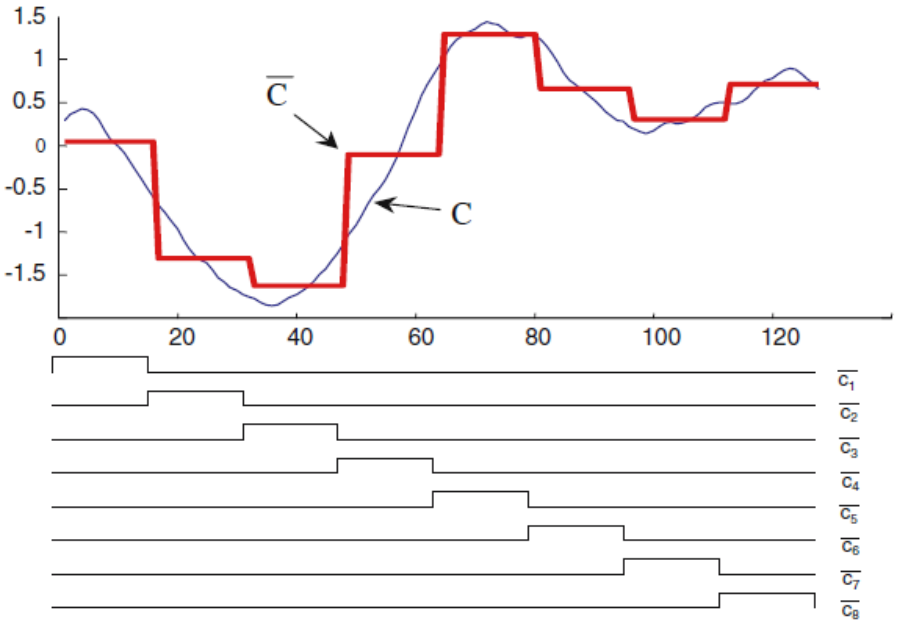

Abbildung 4.1: Beispiel: Piecewise Aggregate Approximation

4.1.3 Test auf Normalverteilung

Für das SAX Verfahren wird eine Normalverteilung der Werte empfohlen. Nur so kann durch die zuvor ausgeführte Z-Normalisierung eine Symbolverteilung von $\frac{1}{a}$ gewährleistet werden. Falls keine Normalverteilung vorliegt, ist die Häufigkeit der Symbole nicht ausgeglichen. Dies stellt für das SAX Verfahren jedoch keine Einschränkung dar, es würde lediglich die Effizienz der anschließenden Lernergebnisse schmälern [5]. Eine Betrachtung der gegebenen Daten zeigt, warum eine z-Normalisierung unumgänglich ist. Die einzelnen

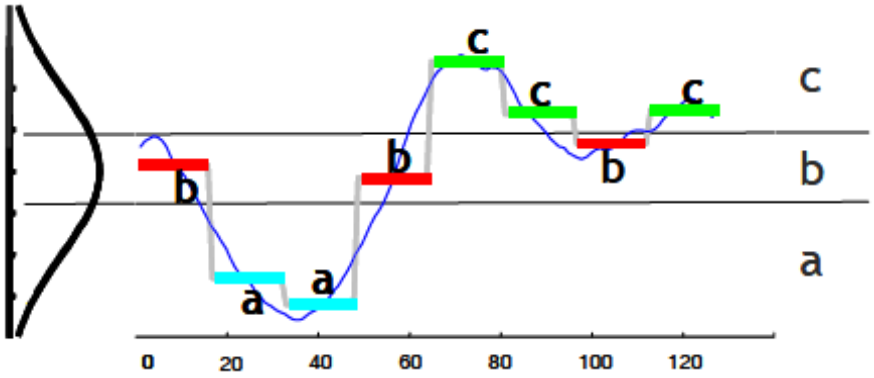

Abbildung 4.2: Beispiel: Symbolüberführung

Walzstiche fangen zeitversetzt an und sind von unterschiedlicher Intensität. Jedoch scheint keine Normalverteilung vorzuliegen. Dies wird durch den *Shapiro-Wilk-Test* aus [21] bestätigt. Dieser ist bereits in dem Programm *R* implementiert worden und kann Wertereihen von 3 bis 5000 Elementen untersuchen. Bei diesem Test wird davon ausgegangen, dass die Daten normalverteilt sind (*Nullhypothese*). Dies wird versucht anhand einer Stichprobe zu widerlegen (*Alternativhypothese*). Da bei diesem Verfahren nur eine Stichprobe betrachtet wird, sollten anschließend die Daten selbst gesichtet werden. Für die 15 Zeitreihen welche eine Länge von über 5000 haben, wurde der *Kolmogorow-Smirnow-Test* aus [11] durchgeführt. Dieser Test ist ebenfalls in *R* implementiert. Dabei werden die Testdaten mit einem normalverteilten Datensatz verglichen. Beide Tests zeigen wie vermutet das keine normalverteilung auf den gegebenen Daten existiert. Auch eine Stichpunktartige Sichtkontrolle der Daten, weist keine Normalverteilung auf, sodass davon der Korrektheit der Testergebnisse ausgegangen wird. Ein Boxplot zweier realer Daten (siehe Abbildung 4.3) im Vergleich zu

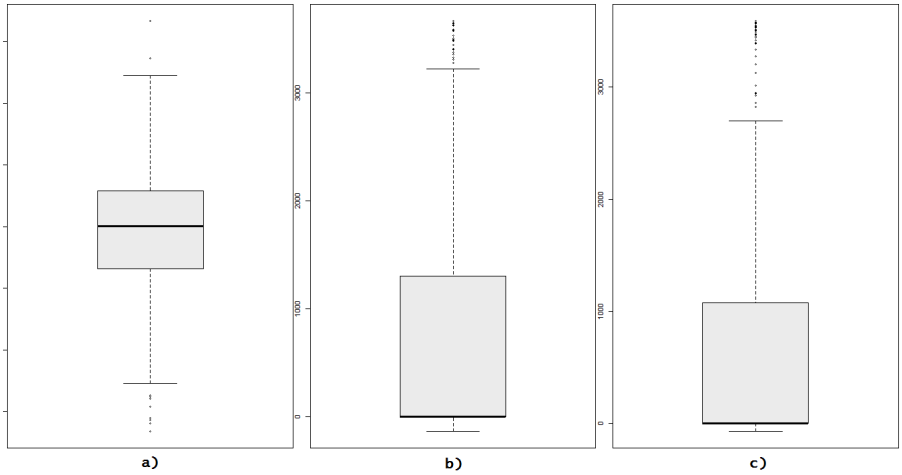

Abbildung 4.3: Boxplot: a) normalverteilter Datensatz b) als gut gelabelter Datensatz c) als schlecht gelabelter Datensatz

einem Normalverteilten Datensatz lässt dies auch leicht erkennen. Der Plot sollte möglichst symmetrisch sein, um nahe an eine Normalverteilung heranzukommen. Wie in a) zu sehen ist der Median (schwarze Linie) die annähernd genaue Mitte. Das bedeutet 50% der Datenelemente liegen drüber und die anderen 50% darunter. In a) verteilen sich diese Daten schön auf den Körper (Boxuntergrenze: 25% Quartil und Boxobergrenze: 75% Quartil) sowie auf die Arme (Whisker). In der realen Datensätzen liegt der Median in etwa bei der Null. Da 50% unter diesem Liegen müssen, verteilen sich 50% der Datenelemente zwischen Null und -20. Die Verteilung in die andere Richtung geht in den 2.000 Bereich. Das bedeutet, in dieser positiven Richtung liegen die Werte viel weiter auseinander als um den Nullpunkt. Aufgrund dessen kann keine Gaußverteilung vorliegen.

4.2 K-Means Clustering

Das k-Means Clustering ist eine weitverbreitete unüberwachte Lerntechnik [3]. Dabei baut das Verfahren eine Aufteilung der gegebenen Daten auf, ohne weitere Informationen über diese zu haben (hier beispielsweise ohne die Zuordnung von guter und schlechter Qualität). Insbesondere kann die Methode keine Resonanz über die Güte der getroffenen Verteilung bekommen, dies bezeichnet man als *Lerner ohne Lehrer*. Die nicht markierten Daten werden k Cluster eingeteilt, sodass die Distanz von den Clusterelementen zu den jeweiligen Clustermittelpunkten(Zentroide) minimal ist. Anschließend werden die Zentroide neu berechnet und der Vorgang wird wiederholt. Falls sich die Clusterzuordnungen nicht mehr ändern terminiert das Verfahren. Der Algorithmus sollte öfters durchlaufen werden, da nur ein lokales Optimum gefunden wird, welches sich bei divergenten Startzentroiden möglicherweise ändert.

4.2.1 k-Means Algorithmus

Einer der ursprünglichen Algorithmen geht auf die 1970iger Jahre von [2] zurück. Zum besseren Verständnis wird meist die Euklidische Distanz als Ähnlichkeitsmaß gewählt. Es sind jedoch auch andere Distanzmaße, der Form $dist : F \times F \rightarrow \mathbb{R}^+$ möglich, welches je zwei Elemente $x, y \in F$ vergleicht und einen Mittelwert dazwischen bilden kann. Beispielsweise hat die später genutzte DTW Distanz diese Eigenschaften.

4.2.1 Definition *k-Means Algorithmus*

1. *Initialisierung: Bilde zufällige Zentroide:* $z_1, ..., z_k$
2. *Wiederhole Folgendes solange* $z_i^t \neq z_i^{t+1}$*, wobei* t *der Schleifendurchlauf ist.*
   (a) *weise jedem Objekt* o *aus der Menge* O*, den nächsten Zentroid* z_i^t *anhand der Distanzfunktion* $dist(o, z_i)$ *zu*

(b) Bestimme neue Zentroide z_i^{t+1}

3. gebe die errechneten Zentroide $z_1, ..., z_k$ *aus*

Wenn dieser Algorithmus durchlaufen ist, wurden k Zentroide, anhand der Zeitreihen generiert.

4.2.2 k-Means als Symbolisierungsverfahren

Durch den k-Means Algorithmus wurde jeder der betrachteten Subsequenzen ein nächster Zentroid zugewiesen. Dies muss nun als Element an die Subsequenz angehangen werden, sodass die genaue Zuordnung $t_i \rightarrow z_j$ besteht. Es kann von der Zeitreihe über die Subsequenz auf die Zentroide geschlossen werden, sodass auch aus der Zeitreihe $T \rightarrow \hat{T}$ gebildet werden kann.

4.3 Symboldarstellung

Nachdem die Symbolisierung abgeschlossen ist, sind die Zeitreihen in die Form $\hat{T} = \hat{t_1}, ..., \hat{t_k}$. Diese können mit einer einfachen Häufigkeitszählung durchlaufen werden, um so diese Zeitreihen für die bekannten Lernverfahren vorzubereiten. Nach einer einfachen Zählung der Symbole (*Monogramm/Unigramm*) befinden sich die Zeitreihen in der Form:

Symbole:	A	B	C	...
ID1	4	0	1	...

Wobei in diesem Beispiel, das Symbol A Vier mal in der gegebenen Zeitreihe vorkommt. Diese Zeitreihen können zu einer gesamten Lerndatentabelle (siehe Abbildung 4.4) zusammengefügt werden. Diese Zählung kann auch mit zwei aufeinanderfolgenden Symbolen (*Bigramm*) oder mehr geschehen. Des Weitern ist eine Überlappung möglich, welche den Symbolverlauf kenntlich macht.

id	1	2	3	4
1001	0.0	9.0	2.0	7.0
1002	0.0	14.0	0.0	1.0
1003	0.0	13.0	1.0	0.0
1004	0.0	11.0	0.0	2.0
1005	0.0	9.0	0.0	5.0
1006	0.0	10.0	3.0	2.0

Abbildung 4.4: Beispiel einer Lerndatentabelle

Kapitel 5

Implementierung in RapidMiner

In diesem Kapitel wird die Implementierung der zuvor vorgestellten Verfahren erläutert. Es beginnt mit den unterschiedlichen Fensterungsmöglichkeiten und setzt sich über die Symbolisierungsverfahren bis zu den Lernern fort. Aufgrund dessen, dass beide Symbolisierungsverfahren bereits als Plug-in des Lehrstuhl 8 zur Verfügung standen, konnten eine große Zahl an Parameter Kombinationen getestet werden, welche in den vorangegangenen Bachelorarbeiten zu kurz kam.

5.1 Datenvorverarbeitung

Bevor mit den Experimenten anhand der B3-Daten begonnen werden kann, müssen die Daten vorbereitet werden. Die Übergabe der B3 Daten erfolgte als Datenpaket der einzelnen Produktionsprozesse, die anhand der Blocknummern zu identifizieren sind. Diese Zeitreihendaten haben eine unterschiedliche Länge, welches einen sofortigen Vergleich nicht zulässt. So wurden verschiedene Techniken angewendet, um die Daten für spätere Experimente vorzubereiten.

5.1.1 Fensterung - statisch

Die wohl einfachste Variante der Fensterung ist die statische. Es wird ein festes Fenster über die Zeitreihe geschoben und es alle so gewonnen Fensterstücke gespeichert. Dieses Sliding Window (siehe Kapitel 3) ist hier noch weiter eingeschränkt, es sieht nur eine nicht überlappende Fensterung vor. Um dies zu verwirklichen, wurde der Subprozess *B3_ Windowing* erstellt. Dieser Subprozess läuft über alle gegebenen Dateien und führt für jede, sich so ergebene Zeitreihe Folgendes aus (siehe Abbildung 5.1).

Zunächst werden die Meta Daten extrahiert und zu einem ID Attribut überführt, welches später wieder an die Daten angehangen wird (unterer Pfad). Die reinen Daten werden zu einem einheitlichen Datentyp verarbeitet und gefenstert (oberer Pfad). Diese Fenste-

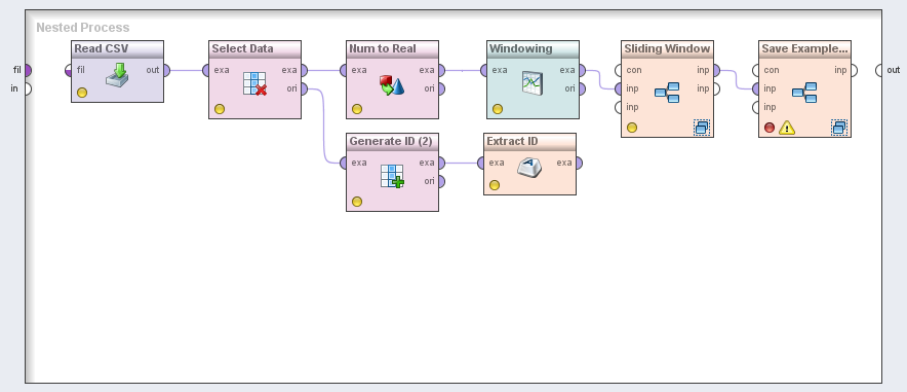

Abbildung 5.1: RapidMiner: B3_Windowing Prozess

rung findet in dem *Windowing* Operator statt, welche mit unterschiedlichen Parametern aufgerufen werden kann. Diese lassen sich von außen, mithilfe von *Macros* definieren und übergeben. Die einfache Fensterung sieht ein nicht Überlappendes vor. Es werden jeweils die einzelnen Fenster in eine neue Zeile überführt, sodass eine einheitliche und vergleichbare Länge der Examples vorhanden ist. In dem letzten Operator werden diese ExampleSets gespeichert und als ein ExampleSet weiter gegeben, welches alle Fenster der gesamten Blockdaten beinhaltet.

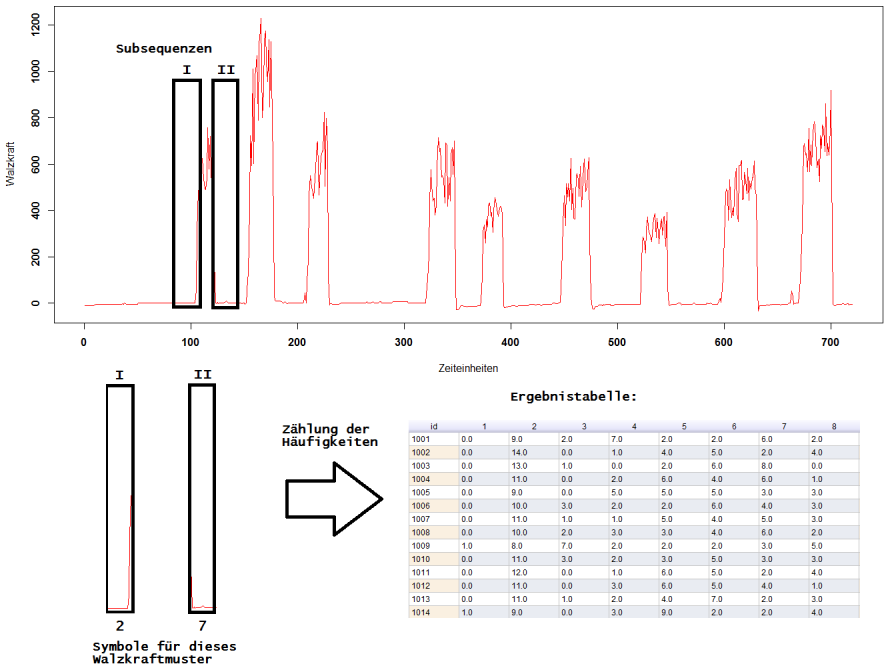

Abbildung 5.2: Konzept der statischen Fensterung

Mithilfe der Macros lässt sich einerseits die Fensteranzahl variieren und andererseits auch ihre Länge. Bei Ungleichheit (Fenstergröße > Schrittlänge) entstehen überlappende Fenster. Da bei den SAX Fenstern nur schwer eine überlappende Realisierung möglich ist, wurde trotz der Möglichkeit keine überlappenden Fenster erstellt.

5.1.2 Fensterung - dynamisch

Durch den Einsatz eines Groovy Scriptes gelang es, zu den statischen, auch dynamische Fenster zu entnehmen. Hierfür wurde ein *Threshold* gewählt um das *vibrieren* der Blockwalze von den eigentlichen Walzstichen zu unterscheiden. Es blieben nur die einzelnen Walzstiche übrig (siehe 5.4). Diese Fenster sind natürlich von ungleicher Länge und wurden als Vorbereitung für das k-Means Clustering gebildet.

Bevor diese Experimente mit den dynamischen Fenstern gestartet worden sind, wurden die untersuchten Fenster aus [8] übergeben. Diese Fenster genossen eine diffizile Vorbereitung. Es wurde ein dynamischer *Threshold* generiert und eine Kantenglättung vorgenommen. Des Weiteren wurden sogenannte Walzanstiche, in einem Fenster gespeichert, welche bei der naiven Vorgehensweise durch zwei Fenster angezeigt wurden. In den folgenden Experimenten wurden die genaueren dynamischen Fenster aus [8] betrachtet und nicht die aus dem obigen Script.

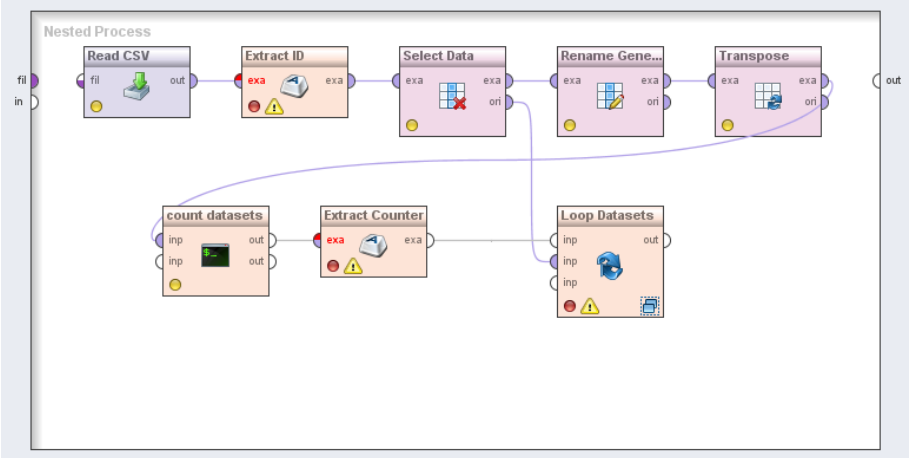

Abbildung 5.3: RapidMiner: B3_Extract_dynamic_Windows Prozess

5.1.3 Symbolisierung mit dem k-Means Clustering

Auf die genannten Fensterungen folgt der k-Means Algorithmus. Hierfür wurden die doppelten Datensätze entfernt und anhand dem übergebenen Parameter *Alphabetgröße* (k) die Zentroide gebildet. Anschließend wurde daraus je ein Modell extrahiert, welches im nächsten Schritt auf den statisch sowie den dynamisch gefensterten Datensatz angewandt worden ist. Dabei wurde jede Subsequenz, dem nächsten Clusterzentroid zugeordnet. Durch

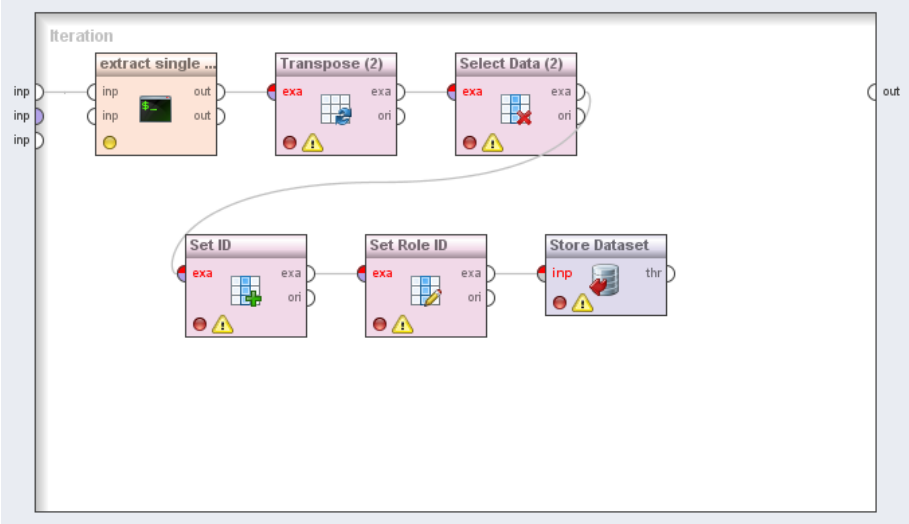

Abbildung 5.4: RapidMiner: B3_Extract_dynamic_Windows_LoopDataSets Prozess

die Zentroid Zuordnung hat bereits die Symbolisierung stattgefunden. Jedem Zentroid wird einem eindeutigen Symbol zugewiesen. Zum Schluss wurde das Qualitätsniveau angefügt, sodass der eigentliche Lerndatensatz vollendet ist.

5.1.4 SAX Fensterung - statisch

Eine andere Möglichkeit ist in dem Plug-in SAX von [12] implementiert worden. Hierbei wird die Zeitreihe in n Intervalle geteilt und in einen neuen Datensatz überführt.

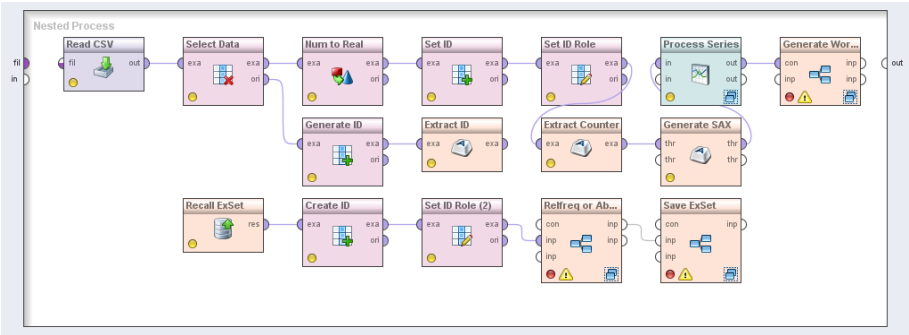

Abbildung 5.5: RapidMiner: B3_Extract_SAX_Windows Prozess

Auch in dem Prozess aus Abbildung 5.5 wird zunächst jede einzelne Datei geladen und verarbeitet. Hier wurde die Möglichkeit offen gelassen die SAX Windows den der normalen Fensterung größtenteils anzupassen. Das noch zu lösende Problem liegt bei Fensterlängen, welche nicht ohne Rest in die Zeitreihen passen. Nachdem die SAX Windows gebildet worden sind, wird die Möglichkeit gegeben, diese in Wörter zu überführen (Operator: *Generate Words*). Zuletzt werden die Daten nochmal geladen um die korrekt ID anzufügen und mittels Hilfsprozessen in die Relativen- oder Absoluten Häufigkeiten zu überführen.

In diesem Prozess kann durch den Parameter der Fensterzahl die größe der einzelnen Fenster passiv bestimmt werden. Des Weiteren kann die größe des Symbolalphabetes und der anschließenden Wortlänge übergeben werden. Bei den Wörtern selbst können überlappende als auch fortlaufende Zählungen angewiesen werden.

5.1.5 Ergebnis der Datenerhebung

Die unterschiedlichen Vorverarbeitungen haben alle dasselbe Ziel (vgl. Lerndatentabelle aus Abbildung 5.6). Zunächst ist die ID eineindeutig auf die Fertigungsnummer zurückzuführen. Die darauf folgende Symbole sind die erhobenen von dem SAX oder k-Means Clustering. Zuletzt folgt als label, die positive oder negative Beurteilung, der Qualität. Die so entstandene Tabelle kann nun zum lernen verwendet werden.

id	2|2	2|9	9|9	6|9	8|8	9|2	3|3	2|8	9|7	Qniveau
1061	36.0	2.0	13.0	1.0	3.0	2.0	0.0	1.0	0.0	true
1062	36.0	2.0	12.0	0.0	5.0	4.0	0.0	1.0	0.0	true
1063	4.0	0.0	10.0	1.0	3.0	2.0	31.0	1.0	0.0	true
1064	29.0	1.0	15.0	1.0	3.0	0.0	0.0	0.0	0.0	true
1065	0.0	0.0	3.0	0.0	0.0	0.0	51.0	0.0	1.0	true
1066	0.0	0.0	11.0	1.0	3.0	1.0	40.0	0.0	0.0	true
1067	35.0	2.0	11.0	0.0	2.0	2.0	0.0	2.0	1.0	true
1068	0.0	0.0	12.0	0.0	3.0	0.0	43.0	0.0	0.0	true
1069	37.0	3.0	13.0	0.0	2.0	0.0	0.0	2.0	1.0	true
1070	29.0	1.0	9.0	1.0	2.0	3.0	0.0	1.0	0.0	true
1071	35.0	2.0	13.0	0.0	3.0	3.0	0.0	0.0	0.0	false
1072	35.0	2.0	9.0	0.0	5.0	1.0	0.0	2.0	0.0	true
1073	35.0	2.0	12.0	0.0	2.0	2.0	0.0	0.0	0.0	true
1074	33.0	0.0	14.0	0.0	4.0	1.0	0.0	1.0	0.0	false
1075	0.0	0.0	8.0	1.0	0.0	0.0	41.0	0.0	0.0	true
1076	33.0	0.0	13.0	0.0	3.0	1.0	0.0	2.0	0.0	true
1077	39.0	3.0	8.0	1.0	1.0	1.0	0.0	0.0	0.0	true
1078	34.0	2.0	13.0	1.0	2.0	4.0	0.0	0.0	0.0	true

Abbildung 5.6: Beispiel einer Lerndatentabelle

5.2 Lernverfahren

Die so entstandenen Datensätze können nun mithilfe von bekannten Lernverfahren bewertet werden. Es wurden die standardisierten Lernen verwendet. Die erstellten RapidMiner Prozesse funktionieren alle nach dem gleichen Prinzip. Zunächst werden alle Lerndatentabellen einzeln geladen und mit einer Kreuzvalidierung bewertet, diese wurde pro Datensatz 10 mal ausgeführt. Als Beispiel wurde der Prozess des Entscheidungsbaumes gewählt (Abbildung 5.7).

In jedem Lernverfahren wurden, falls möglich, die Parameter optimiert. Hierfür wurde die *Grid Optimization* genutzt. Nachdem die optimalen Parameter in gesuchten Reichweite gefunden wurde, konnte jeder Datensatz anhand seiner Präzision (im Folgenden *accuracy*), sowie des *weighted-mean-recall* und *weighted-mean-precision* bewertet werden. Diese wurden dann mitsamt der Optimalen Parameter extrahiert, sodass eine Überprüfung der Ergebnisse möglich ist. Des Weiteren wurde die Konfidenz Tabelle angehangen.

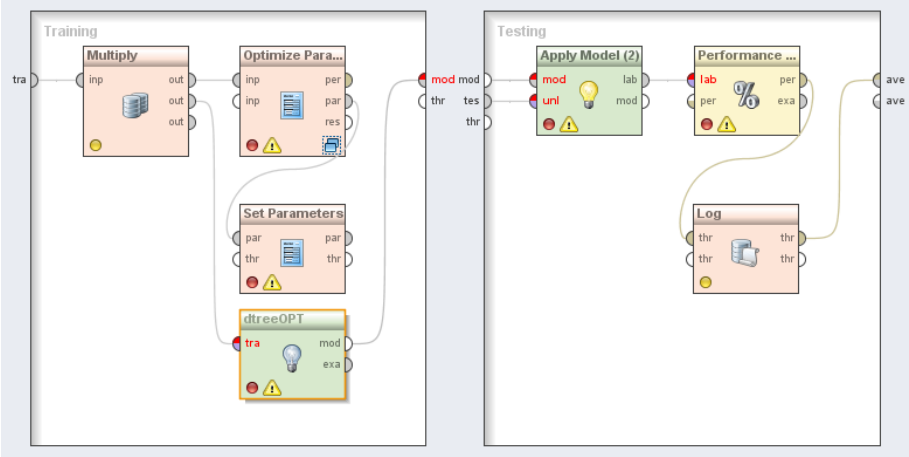

Abbildung 5.7: Decision Tree Lerner

Dieser Entscheidungsbaum Lerner zeichnet sich durch seine Vielzahl an Parametern aus. Jedoch konnten nicht alle Datensätze korrekt getestet werden, da es immer wieder Probleme bei der Berechnung der *normalInverse()* von RapidMiner gab. Durch den Operator, *Handle Exception* konnten zwar Ergebnisse extrahiert werden, aber deren Korrektheit ist zweifelhaft. Bis zum Abschluss der Arbeit wurde der Grund nicht aufgeklärt, weshalb nur die beiden anderen Lernergebnisse zur Verfügung stehen.

Kapitel 6

Experimente

Die zuvor dargestellten Symbolisierungsverfahren, sind mit in die Prozesse von RapidMiner integriert worden. Diese können mit unterschiedlichen Parametern wie: Fensteranzahl, Symbolalphabet, Wortlänge etc. viele divergente Datensätze generieren. Die Erläuterung der generierten Datensätze wird in dem ersten Teil dieses Kapitels vorgestellt. Anschließend folgt die von den Lernern mögliche Parameter Optimierung, welche ebenfalls von RapidMiner automatisiert vorgenommen wird. Die einzustellenden Richtwerte sind mit einer kurzen Erklärung, der Übergang zu den erhobenen Ergebnissen. Bevor diese diskutiert werden, wird ein Maß definiert. Der Default-Lerner ist der jetzige Zustand und diesen gilt es zu übertreffen. Zuletzt werden die Lernergebnisse in verschiedenen Abbildungen dargestellt und interpretiert.

6.1 Parameter Kombinationen

Wie in Kapitel 5 bereits beschrieben, wurden die B3 Daten unterschiedlichst aufbereitet und anschließend mit Standard Lernverfahren evaluiert. Aus den Rohdaten sind durch die Vorverarbeitung unterschiedlichste Datensätze entstanden, die Parameter Kombinationen sind im Folgenden aufgelistet:

- dynamische Fenster:
  - Symbolalphabet: 2 bis 30 → *29 Möglichkeiten*
  - Gesamt: **29 Lerndatensätze**
- statische Fenster:
  - Symbolalphabet: 2 bis 30 → *29 Möglichkeiten*
  - Fenstergröße: 5 bis 50 in 5*er* Schritten → *10 Möglichkeiten*
  - Wortlänge: 1 bis 4 → *4 Möglichkeiten*

  - überlappende Wörter: *ja* oder *nein* → *2 Möglichkeiten*
  - Gesamt: **2030 Lerndatensätze**[1]

- statische SAX Fenster:
  - Symbolalphabet: 2 bis 15 → *14 Möglichkeiten*
  - Fensteranzahl: 30 bis 150 in 10er Schritten → *13 Möglichkeiten*
  - Wortlänge: 1 bis 4 → *4 Möglichkeiten*
  - überlappende Wörter: *ja* oder *nein* → *2 Möglichkeiten*
  - Gesamt: **1274 Lerndatensätze**[2]

Das SAX Verfahren hat eine geringer Alphabetgröße da, bei einem zu großen Alphabet die meisten Subsequenzen auf einzigartige Wörter abgebildet werden würden, welches zu keinem guten Lernergebnis führt [5]. Dort wird eine Größe von Drei oder Vier als ausreichend für *virtually any task on any dataset* angegeben. Die entsprechenden Lernverfahren wurden mit jedem dieser Datensätze aufgerufen. Dabei wurden folgenden Parameter Kombinationen für die jeweiligen Lernverfahren angedacht:

- Naive Bayes:
  - RapidMiner bietet keine Parameter Optimierung für dieses Lernverfahren an
- k-Nächste-Nachbarn:
  - k = 3 bis 101 → 50 Kombinationen
- Decision Tree:
  - minimal size for split = 1 bis 30 → 11 Kombinationen
  - minimal leaf size = 1 bis 75 → 11 Kombinationen
  - maximal depth = 2 bis 10 → 5 Kombinationen)
  - number of prepruning alternatives = 2 bis 10 → 5 Kombinationen
  - minimal gain = 0.001 bis 1 → 6 Kombinationen
  - confidence = 0.25 bis 1 → 4 Kombinationen
  - Insgesamt **72.600** Kombinationen

[1] Rechnerisch 2320 Datensätze, aber bei überlappenden Wörtern, mit Wortlänge = 1 entsteht ein identischer Lerndatensatz

[2] Rechnerisch 1456 Datensätze

6.2 Ergebnisse der Lerndaten

Im folgenden werden die erworbenen Erkenntnisse aus den verschiedenen Lernverfahren dargestellt. Dabei wird zunächst der derzeitige Richtwert (Default Lerner) ermittelt und anschließend dahingehend die Ergebnisse untersucht.

6.2.1 Zielergebnis

Bevor die Lernergebnisse bewertet werden könne, wird eine Richtnorm benötigt, an dessen diese Ergebnisse gemessen werden. In diesem Fall ist der IST-Zustand der Default Lerner. Alle Produkte werden als positiv bewertet, was bedeutet das genau die Fehlerhaften Produkte auch die Fehlerrate ist. Die Wahrscheinlichkeit, dass ein Produkt wirklich korrekt ist, nach der positiven Bewertung lässt sich wie folgt bestimmen:

$$P(korrekten\_Klassifizierung) = \frac{richtig\_klassifiziert}{gesamt} = \frac{344}{372} = 0,9247 = 92,47\% \tag{6.1}$$

Zur Veranschaulichung wird eine Koeffizienz Matrix angegeben. Für den Default Lerner ist dies in der folgenden Tabelle angegeben. Dabei zeigen die Reihen (*negativ-, positiv bewertet*) die jeweilige Einschätzung an, wobei in der letzten Spalte die Gesamten negativ- und positiv bewerteten Elemente stehen. In den Spalten, wahr und falsch, werden die korrekt Bewerteten Elemente eingetragen, in dem vorliegenden Fall: 28 mal falsch und 344 mal wahr.

Koeffizienz Matrix:	falsch	wahr	Gesamt
negativ bewertet	$f_n = 0$	w_n=0	$f_n + w_n = 0$
positiv bewertet	$f_p = 28$	w_p=344	$f_p + w_p$=372

Um ein besseres Ergebnis als der Default Lerner zu bekommen, müsste wie Folgt von dem Zustand abgewichen werden, indem mehr negativ-falsch bewertet würden als negativ-wahr. Jedoch spiegelt dies nur die Genauigkeit der Vorhersage wieder. Die folgende Koeffizienz Tabelle zeigt die Diskrepanz:

Koeffizienz Matrix:	falsch	wahr	Gesamt
negativ bewertet	$f_n = 28$	w_n=27	$f_n + w_n = 55$
positiv bewertet	$f_p = 0$	w_p=317	$f_p + w_p$=317

$$P(korrekten\_Klassifizierung) = \frac{(f_n+w_p)}{gesamt} = \frac{345}{372} = 92,74\%$$

Das obige Ergebnis wäre marginal besser als der Default Lerner, jedoch angewandt auf das zu lösende Problem, um ein vielfaches schlechter. Der Default Lerner lässt alle Produkte bis zur Qualitätskontrolle und verwirft 28 mal das Produkt. Es müssten erneut diese 28 Produkte hergestellt werden. Der Einfachheit halber wird angenommen, dass bei einer zweit Produktion immer fehlerfreie Produkte gefertigt werden. Bei der zweiten und

genaueren Klassifizierung werden insgesamt 55 Produkte während der Produktion verworfen. Es werden zwar alle als fehlerfrei klassifizierten Produkte auch die Qualitätsprüfung bestehen, jedoch müssten 55 neue Produkte hergestellt werden. Dies bedeutet, dass die negativ-wahren Elemente sehr viel gravierender sind, als die positiv-falschen. Dieses Problem ist bereits ein bekanntes in der Informatik, beispielsweise bei einem E-Mail Spam Filter. Falsch-negative Emails werden schnell gelöscht, jedoch wie häufig werden wahr-negative Mails aus dem Spam Ordner gelesen? Dies gilt es in den folgenden Experimenten mit zu beachten. Dazu sei noch gesagt, dass der Default Lerner bereits das Optimum darstellt, welches den Fehler der negativ-wahren Elemente minimiert. Es existiert keine Kombination, welche eine geringere Genauigkeit bei minimaleren Fehler dieser Art zulässt. Infolgedessen reicht es die Ergebnisse über der Default Lerner Genauigkeit zu betrachten.

6.2.2 Ergebnisse

Die Ergebnisse der einzelnen Lernverfahren, sind leider nicht so aussagekräftig ausgefallen wie erhofft. Nur selten wurde der Default Lerner übertroffen. Wenn ein Lernverfahren auf demselben Niveau wie der Default Lerner liegt, wird dies im Folgenden als Durchschnittlich bezeichnet. Die Untersuchung der einzelnen Stiche ergab, mit dem Naive Bayes Klassifikator, bei vier Alphabetsgrößen eine Verbesserung zu dem Default Lerner. Es wurde jeweils bei k=2, k=3, k=8 und k=18 ein Fehlprodukt erfolgreich als fehlerhaft gekennzeichnet, sonst bewegte sich dieser Datensatz auf höhe des Default Lerners.

k-Means Stiche k=3:	falsch	wahr	Gesamt
negativ bewertet	$f_n = 1$	w_n=0	$f_n + w_n = 1$
positiv bewertet	$f_p = 27$	w_p=344	$f_p + w_p$=371

$$P(korrekten\_Klassifizierung) = \frac{(f_n+w_p)}{gesamt} = \frac{345}{372} = 92,74\%$$

Bei dem k-Means Clustering wurde etwas sehr interessantes festgestellt. Durch einen Fehler wurde dieser Datensatz auch mit einem Alphabet k = 1 erstellt. Dies bedeutet es wäre trotz allem Clustern, die Zeitreihe nur auf eine 1 abgebildet. Zwei dieser statischen Datensätze erzielte ohne Variation in den Symbolen, ein erstaunliches Ergebnis von 93,28% richtig klassifizierten(k-NN) Zeitreihen. Bemerkenswert ist dies, da in diesem Datensatz einzig die Länge das ausschlaggebende Merkmal sein kann.

k-Means Clustering k=1:	falsch	wahr	Gesamt
negativ bewertet	$f_n = 4$	w_n=1	$f_n + w_n = 5$
positiv bewertet	$f_p = 24$	w_p=343	$f_p + w_p$=367

$$P(korrekten\_Klassifizierung) = \frac{(f_n+w_p)}{gesamt} = \frac{347}{372} = 93,28\%$$

Einer der besten Lerndatensatze ist aus dem nicht überlappenden k-Means Clustering entstanden (siehe Abbildung 6.1). Das Ergebnis von 93,55% liegt jedoch kaum höher als

der Default Lerner. Diese Schwierigkeit diesen zu übertreffen, war jedoch von Anfang an klar.

k-Means Clustering:	falsch	wahr	Gesamt
negativ bewertet	$f_n = 6$	w_n=2	$f_n + w_n = 8$
positiv bewertet	$f_p = 22$	w_p=342	$f_p + w_p$=364

$$P(korrekten\_Klassifizierung) = \frac{(f_n+w_p)}{gesamt} = \frac{348}{372} = 93,55\%$$

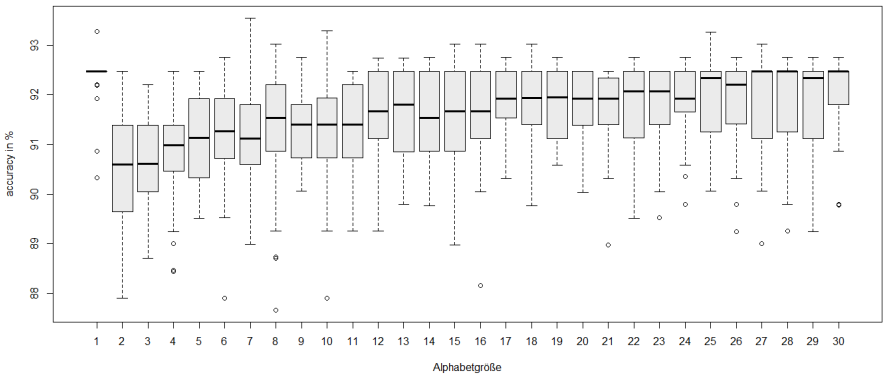

Abbildung 6.1: Ergebnis: Symbolisierung mit k-Means Clustering Klassifizierung mit k-NN

Die Wahl der richtigen Alphabetgröße (Zentroidanzahl) scheint auch bei dem k-Means Clustering eine höher geordnete Rolle zu spielen. Obwohl auch Datensätze mit einem großen Alphabet gute Lernergebnisse erzielen können, fällt der Durchschnitt ab. Sehr gut zu erkennen ist dies in Abbildung 6.2. Dort fällt der Median ab einem k = 22 auf etwa 70%.

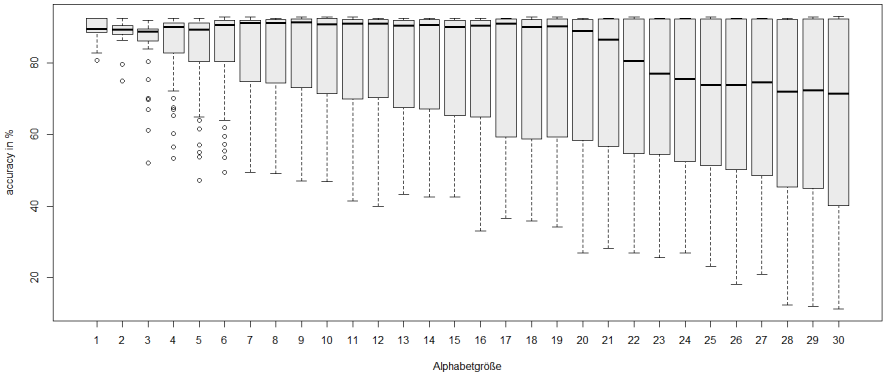

Abbildung 6.2: Ergebnis: Symbolisierung mit k-Means Clustering Klassifizierung mit Naive Bayes

Die SAX Symbolisierung erreicht in allen Bereichen Durchschnittliche Ergebnisse, also auf Höhe des Default Lerners. Nur in einem Fall wird dieser deutlich übertroffen (SAX überlappende Fenster mit k-NN). Hier werden dieselben Klassifizier-Ergebnisse erreicht wie beim besten des Clusterings.

k-Means Clustering:	falsch	wahr	Gesamt
negativ bewertet	$f_n = 4$	w_n=0	$f_n + w_n = 4$
positiv bewertet	$f_p = 24$	w_p=344	$f_p + w_p$=368

$$P(korrekten\_Klassifizierung) = \frac{(f_n+w_p)}{gesamt} = \frac{348}{372} = 93,55\%$$

Dennoch ist dies das bessere Ergebnis bei dem vorliegenden Anwendungsfall. Es konnten insgesamt vier Produkte korrekt aussortiert werden, ohne dabei ein möglich gutes Produkt zu verwerfen. Bei dem k-Means Clustering konnten sechs korrekt klassifizierte und zwei falsch klassifizierte verworfen werden. In diesem Fall würden 8 Produkte entfernt, wobei zwei eigentlich Fehlerfrei seien.

6.2.3 Zusammenfassung

Trotz der Masse an erstellten Daten konnten kaum bessere Ergebnisse als der Default Lerner gewonnen werden. Die einzig zwei brauchbaren Ergebnisse wurden je vom k-Means Clustering als auch von der SAX Symbolisierung erzielt:

- SAX überlappende Wörter, Alphabet=12, Fensteranzahl=150, Wortlänge=4 → Naive Bayes: 93.55%
- k-Means nicht überlappende Wörter, Alphabet=7, Fenstergröße=5, Wortlänge=1 → k-NN: 93.55%

Beide Verfahren nutzen die Minimale Fensterlänge aus. Bei dem SAX werden insgesamt 150 Subsequenzen aus einer Zeitreihe extrahiert, bei dem k-Means haben diese Subsequenzen eine Länge von 5. Des Weiteren ist Interessant zu sehen, dass obwohl eine Symbolanzahl größer als vier nicht üblich ist, dass SAX erst mit 12 Symbolen die Bestleistung erbracht hat. Auch könnte die überlappende Wortlänge von vier als Kompensation des Formverlustes betrachtet werden. Da SAX nur die Mittelwerte bestimmt und so keine Information über den Verlauf der Zeitreihen hat, kann durch eine beispielsweise aufsteigende Symbolanordnung der Verlauf nachgebildet werden.

Kapitel 7

Fazit

In dieser Arbeit wurden reale Daten eines Stahlherstellers untersucht, um dessen Produktionsprozess zu optimieren. Das Problem der hohen Fehlkosten in der Stahlindustrie ist bekannt und sucht weiterhin nach einer Lösung. Die Kontrolle zwischen den Produktionsstationen ist sehr unpraktikabel, da das Produkt meist erst abkühlen müsste und so ein weiteres Produktionsgut verbraucht wird, die Zeit. Aufgrund dessen besteht der Versuch von Sensordaten der Produktion auf die Qualität des Endproduktes zurück zu schließen. Diese Qualitätseinstufung wurden mithilfe von zwei Symbolisierungsverfahren untersucht. Da eine direkte Überprüfung nicht möglich war, mussten die Daten vorbereitet werden. Für das bekannte Verfahren SAX wurde die Fensterung des Plug-ins von [12] in RapidMiner übernommen. Dabei konnten aus den übertragenen Rohdaten insgesamt **1.274** Datensätze gewonnen werden. Diese stellt sich ausverschiedensten Parameter Kombinationen zusammen: Fensteranzahl, Symbolalphabet, Wortlänge(mit und ohne Überlappung). Die Ergebnisse waren wie in allen Versuchen eher ernüchternd. Das beste Lernergebnis (93,55%), ist kaum besser als der Default Lerner. Jedoch kann für weiterführende Arbeiten Folgendes mitgenommen werden. Die Symbolanzahl sollte nicht minimiert werden, sondern auch hinreichend groß gewählt werden. Während bei dem k-Means Clustering ein abfallen, der Klassifizierungsergebnisse bei einem k=22 zu erkennen war, ließ die Güte von SAX bis zu den hier maximal betrachteten k=15 nicht nach. Jedoch hat die SAX Symbolisierung nur mit einer überlappenden Wortlänge von Vier dieses Ergebnis erzielt, möglicherweise könnte die Wortlänge für eine Verbesserung erhöht werden.
Die andere untersuchte Möglichkeit der Symbolisierung, das k-Means Clustering, erzielte auch eine Genauigkeit der Klassifizierung von 93,55%. Dabei wurde die RapidMiner Extension aus [15] genutzt, um mit dem DTW Distanzmaß die Zeitreihen auf Ähnlichkeit zu untersuchen. Als Erstes wurden auch hier die Rohdaten in **2.030** Datensätze zerlegt. Trotz der gleichen Länge wurde auf das einfache Distanzmaß der Euklidischen Distanz verzichtet. So konnte der Informationsvorteil besser verglichen werden. Während das SAX lange überlappende Fenster brauchte, konnte das k-Means Clustering ohne dies das beste Lern-

ergebnis erzielen. Beide Verfahren nutzten die maximal Anzahl an Subsequenzen, sodass auch hier möglicherweise noch Performanz Optimierungen möglich sind, indem kleinere Subsequenzlängen gebildet werden.
Der zuletzt untersuchte Datensatz, aus [8], konnte aus zeitlichem Mangel nur grob untersucht werden. Hierbei wurden das Symbolalphabet verändert, sodass **29** Datensätze aus den dynamischen Fenstern entstanden sind. Die vorherige Option das Euklidische Distanzmaß zu wählen bestand somit nicht mehr und es musste die DTW Distanz in Verbindung mit dem k-Means Clustering genutzt werden. Da die einzelnen Fenster eine Länge von etwa 30 bis 170 aufwiesen, könnten auch hier wiederum Subsequenzen gebildet werden, was eine Optimierung auf diesem Wege ebenfalls möglich erscheinen lässt.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Deuse, Jochen, Benedikt Konrad, Daniel Lieber, Katharina Morik und Marco Stolpe: *Challenges for data mining on sensor data of interlinked processes.* Proc. of the Next Generation Data Mining Summit 2011: Ubiquitous Knowledge Discovery for Energy Management in Smart Grids and Intelligent Machine-to-Machine (M2M) Telematics, 2012.

[2] Hartigan, John A und Manchek A Wong: *Algorithm AS 136: A k-means clustering algorithm.* Applied statistics, Seiten 100–108, 1979.

[3] Hastie, Trevor, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman und R Tibshirani: *The elements of statistical learning.* Springer, 2001.

[4] Keogh, Eamonn und Shruti Kasetty: *On the need for time series data mining benchmarks: a survey and empirical demonstration.* Data Mining and knowledge discovery, 7(4):349–371, 2003.

[5] Keogh, Eamonn, Jessica Lin und Ada Fu: *Hot sax: Efficiently finding the most unusual time series subsequence.* In: *Data mining, fifth IEEE international conference on*, Seiten 8–pp. IEEE, 2005.

[6] Keogh, Eamonn und Chotirat Ann Ratanamahatana: *Exact indexing of dynamic time warping.* Knowledge and information systems, 7(3):358–386, 2005.

[7] Kim, Sang-Wook, Sanghyun Park und Wesley W Chu: *An index-based approach for similarity search supporting time warping in large sequence databases.* In: *Data Engineering, 2001. Proceedings. 17th International Conference on*, Seiten 607–614. IEEE, 2001.

[8] Lieber, Daniel, Marco Stolpe, Benedikt Konrad, Jochen Deuse und Katharina Morik: *Quality prediction in interlinked manufacturing processes based on supervised & unsupervised machine learning.* Procedia CIRP, 7:193–198, 2013.

[9] Lin, Jessica, Eamonn Keogh, Stefano Lonardi und Bill Chiu: *A symbolic representation of time series, with implications for streaming algorithms.* In: *Proceedings*

of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, Seiten 2–11. ACM, 2003.

[10] Lin, Jessica, Eamonn Keogh, Li Wei und Stefano Lonardi: *Experiencing SAX: a novel symbolic representation of time series.* Data Mining and Knowledge Discovery, 15(2):107–144, 2007.

[11] Massey Jr, Frank J: *The Kolmogorov-Smirnov test for goodness of fit.* Journal of the American statistical Association, 46(253):68–78, 1951.

[12] Matuschek, Christian: *Symbolisierung und Clustering von Zeitreihen als neue Operatoren im ValueSeries Plugin von Rapidminer.* 2013.

[13] Mierswa, Ingo: *Automatisierte Merkmalsextraktion aus Audiodaten.* LS8 - TU Dortmund, Diplomartbeit, 2004.

[14] Müller, Meinard: *Information retrieval for music and motion*, Band 2. Springer, 2007.

[15] Pfahler, Lukas: *Effizienteres k-means Clustering von Zeitreihen mit Dynamic Time Warping durch kaskadiertes Berechnen von unteren Schranken.* 2013.

[16] Rakthanmanon, Thanawin, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria und Eamonn Keogh: *Searching and mining trillions of time series subsequences under dynamic time warping.* In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Seiten 262–270. ACM, 2012.

[17] Reppel, Niklas, Katharina Morik und Dipl-Inform Marco Stolpe: *Entwicklung eines Annotierungswerkzeuges für Musikdaten.*

[18] Sakoe, Hiroaki und Seibi Chiba: *Dynamic programming algorithm optimization for spoken word recognition.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 26(1):43–49, 1978.

[19] Schlittgen, Rainer und Bernd HJ Streitberg: *Zeitreihenanalyse.* Oldenbourg Verlag, 2001.

[20] Senin, Pavel und Sergey Malinchik: *SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model.* Seiten 1175–1180, 2013.

[21] Shapiro, Samuel Sanford und Martin B Wilk: *An analysis of variance test for normality (complete samples).* Biometrika, Seiten 591–611, 1965.

[22] Witten, Ian H und Eibe Frank: *Data Mining: Practical machine learning tools and techniques.* 2001.