technische universität
dortmund

Bachelorarbeit

Symbolisierung und Clustering von Zeitreihen als neue Operatoren im ValueSeries Plugin von Rapidminer

Christian Matuschek
September 2013

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

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (VIII)
http://ls8-www.cs.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Einleitung und Zielsetzung

Das Teilprojekt B3 des *Sonderforschungsbereich 876 - Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung* befasst sich mit Data Mining in Sensordaten automatisierter Prozesse. Eine Qualitätsprognose noch während des Verarbeitungsprozesses soll Fehlproduktionen möglichst früh erkennen um so Mehrkosten zu vermeiden. Als konkrete Fallstudie dienen Daten verschiedener Sensoren aus einem Stahlwalzwerk. Dazu gehören Temperaturdaten der gewalzten Stahlblöcke, die angewandte Walzkraft und die Drehzahlanstellung der Walzmaschine.

In bisherigen Versuchen Qualitätsprognosen zu erstellen wurde eine große Menge manuell ausgewählter, lokaler und globaler Merkmale extrahiert. Globale Merkmale heißen Eigenschaften, die über den gesamten Walzvorgang gelten. Ein Walzvorgang teilt sich in mehrere Durchläufe des Materials durch die Walze, auch *Stiche* genannt, auf. Die Menge an Stichen pro Walzvorgang schwankt zwischen 3 und 6 Stichen. Lokale Merkmale beziehen sich jeweils nur auf Eigenschaften eines Stichs.

Um Merkmale heraus zu filtern, welche Aufschluss über die zu erwartende Qualität des Endproduktes geben, wurde Versucht, die Menge aller extrahierten Merkmale durch ein evolutionären Algorithmus zu optimieren. So wurde eine Vorhersagegenauigkeit von 80,21% erreicht.

Um die Vorhersagegenauigkeit weiter zu steigern, werden in dieser Arbeit Vorverarbeitungsmethoden des RapidMiner ValueSeries Plugins angewandt. Die Anzahl von möglichen Vorverarbeitungsprozessen steigt so enorm an. Zur Suche nach einem möglichst optimalen Vorverarbeitungsprozess, wird ein evolutionäres Optimierungsverfahren angewandt [20] [22]. Dieses Optimierungsverfahren wurde bereits zur automatischen Merkmalsextraktion von Audiodateien implementiert und führte bei dieser Aufgabenstellung zu guten Ergebnissen [20].

Die Methoden des ValueSeries Plugins umfassen typische Methoden zur Analyse von periodischen, sinusoiden Zeitreihen wie Audiodaten in Wellenform (bsp. Fourier Transformation). Für Daten mit komplexe ren Dynamiken sind diese nur bedingt geeignet [24].

Daher wird versucht, durch Symbolisierung, Bezug auf die Form der gemessenen Zeitreihen zu nehmen.

Es werden zwei verschiedene Symbolisierungsverfahren in das ValueSeries Plugin integriert. Die entstehenden Symbolreihen fassen die Form der Zeitreihen geglättet und komprimiert zusammen und ermöglichen die Anwendung zusätzlicher Verfahren aus dem Bereich des Data Mining. Des weiteren kann die Symbolreihe als direkter Input eines Lernverfahrens genutzt werden ohne dabei hohe Laufzeit zu fordern.

Bei den Symbolisierungsverfahren handelt es sich um ***S****ymbolic* ***A****ggregate appro****X****imation* (kurz SAX) und der Symbolisierung durch Clustering.

SAX übersetzt die Zeitreihe in eine Symbolreihe und stellt ein Distanzmaß auf dieser zur Verfügung. Bei der Symbolisierung durch Clustering wird jede Zeitreihe anhand einem Cluster zugeordnet. Ein Cluster ist eine Gruppierung von Objekten, die möglichst ähnlich zueinander sind. Durch die Symbolisierung ist es möglich die Zeitreihe direkt als Eingabe eines Lernverfahrens zu verwenden und so Zeitreihen ressourceneffizient zu vergleichen. Weiterhin werden aus den Symbolreihen weitere Merkmale extrahiert die sich möglicherweise für die Klassifikation zur Qualitätsprognose eignen.

Anschließend sollen verschiedene Experimente Aufschluss darüber geben, wie die durch Symbolisierung erweiterte automatische Merkmalsextraktion die Vorverarbeitung, hinsichtlich der Klassifikation von Wertereihen, verbessert. Dazu werden mehrere Experimente mit der Data Mining Software RapidMiner vorbereitet und auf Messdaten der Fallstudie des Stahlwalzwerkes angewandt.

Im folgenden Kapitel werden die zu integrierenden Symbolisierungsverfahren vorgestellt. Kapitel 3 geht auf die Merkmalsextraktion mittels evolutionären Algorithmen ein. Kapitel 4 Beschreibt die Integration der Verfahren in RapidMiner und Kapitel 5 zeigt die Evaluation dieser. Schließlich werden die Ergebnisse dieser Arbeit in Kapitel 6 zusammengefasst und es wird ein Ausblick auf mögliche Erweiterungen der Verfahren gegeben.

Kapitel 2

Symbolisierung von Zeitreihen

Bisher extrahierte Merkmale beschreiben Eigenschaften der gesamten Sensordaten, bzw Eigenschaften eines Stichs. Es ist jedoch denkbar, dass lokale Eigenschaften, wie die Form der Zeitreihe Aufschluss über die zu erwartende Qualität des Walzmaterials geben kann. Beispielsweise wenn die Walzkraft zu Beginn des Walzvorganges zu stark eingestellt ist und das Material so Schaden nimmt.

Die Symbolisierung von Zeitreihen bringt drei Vorteile gegenüber anderen Methoden der Zeitreihenanalyse mit sich: Durch Segmentierung der Zeitreihe und Umsetzung der Segmente in ein Symbol, kann sich die Laufzeit darauf folgender Analysemethoden deutlich verkürzen [9]. Dabei bleibt die Form der Zeitreihe erhalten und fließ in den Analyseprozess mit ein. Wie stark die Daten komprimiert werden, hängt von der Wahl der Größe der Segmente ab. Grobe Segmentierung fasst die Zeitreihe in weniger Symbolen zusammen, vergrößert aber auch den Verlust von Informationen. Weiterhin glättet die Symbolisierung die Zeitreihe und reduziert so Anfälligkeit, Rauschen und Fehlmessungen [9].

Zuletzt erlaubt die Umsetzung von Zeitreihen in Symbolreihen die Nutzung von Analysemethoden aus gut erforschten Bereichen wie Data Mining in Datenbanken, Text-Mining, Wissenrepräsentation & Wissensverarbeitung, Computerlinguistik und Bioinformatik [24], [19], [16].

Im Folgenden werden zwei Symbolisierungsverfahren vorgestellt. SAX hat sich als Standard zur Symbolisierung von Zeitreihen etabliert. SAX sticht aus der Menge von Symbolisierungsverfahren dadurch heraus, dass es erlaubt ein Distanzmaß auf der symbolischen Repräsentation der Daten zu definieren. Ein Distanzmaß definiert eine Funktion $d(x, y)$ auf allen Objekten x und y einer Menge, welche die Ähnlichkeit zwischen beiden Objekten beschreibt. So lassen sich mit dem SAX Distanzmaß Distanzen zwischen Symbolen berechnen. Dies erlaubt den Einsatz von verschiedenen Data Mining-Algorithmen direkt auf der symbolischen Repräsentation, ohne die original Daten referenzieren zu müssen. Bei reduzierter Laufzeit erzielt man so die gleichen Ergebnisse, wie bei Anwendung der Data Mining-Verfahren auf den original Daten [19].

Diskretisierung von Zeitreihen durch Clustering ist eine gebräuchliche Methode [10], [15], [12]. Dem Clusteralgorithmus werden mehrere Zeitreihen oder Teilstücke von Zeitreihen der Länge n übergeben. Dieser behandelt die Zeitreihen als n-Dimensionale Beispiele und ordnet sie einem von k Clustern zu. k muss dabei als Parameter gesetzt worden sein. Schließlich werden die Zeitreihen durch ihre Clusterzuordnung ersetzt. Diese können als diskrete Symbolwerte betrachtet werden.

2.1 Verwandte Arbeiten

Für die Verarbeitung von Zeitreihen gibt es mehrere Darstellungsarten. Dazu zählen neben der symbolischen Darstellung durch SAX, Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Piecewise Linear und Piecewise Constant Models(PAA) und Singular Value Decomposition (SVD) [19]. Diese Darstellungsformen haben jeweils Vor- und Nachteile. Gemein ist allen Darstellungsarten jedoch, dass ihre Werte aus Gleitkommazahlen bestehen. Durch die diskrete, symbolische Darstellung ermöglicht SAX die Anwendung einer vielzahl von Methoden, beispielsweise des Zählens von gleichen Teilabschnitten einer Zeitreihe (engl.: Frequent Item(sets)).

Weitere Verfahren zur Symbolisierung von Zeitreihen sind ***A**ligned **C**luster **A**nalysis* [26] und *Persist* [21]. Aligned Cluster Analysis (ACA) eignet sich speziell für menschliche Bewegunsdaten, wobei SAX sich gut für unperiodische Daten eignet, wie sie in der Fallstudie des Stahlwalzwerkes hauptsächlich vorliegen [24].

Aus diesen Gründen, und weil SAX sich als Standard für die symbolische Darstellung von Zeitreihen durchgesetzt hat, wird in dieser Arbeit SAX in die automatische Merkmalsextraktion integriert. Außerdem wird versucht, Zeitreihen durch Clusterung mittels bereits implementierten Clusteringalgorithmen des RapidMiner zu symbolisieren.

2.2 SAX

SAX steht für ***S**ymbolic **A**ggregate appro**X**imation* und beschreibt eine Methode zur Symbolischen Repräsentation von Zeitreihen. SAX wurde von Eamonn Keogh vom Department of Computer Science and Engineering der University of California, Riverside und Jessica Lin vom Information and Software Engineering Department der George Mason University entwickelt.

SAX zeichnet sich durch zwei nützliche Eigenschaften aus: Es kann ein Distanzmaß auf der symbolischen Repräsentation der Daten definiert werden, dessen Werte stark mit denen der original Daten korrelieren. Außerdem glättet und komprimiert SAX die Länge der Zeitreihe nach belieben und kann so die Performanz bei nachfolgenden Verfahren erheblich verbessern. Die Kompression erfolgt durch Fensterung und Übersetzung der Fenster in

Symbole, die Glättung durch Bildung des Mittelwertes über jedes Fenster. Kompression und Genauigkeit des Distanzmaßes können durch Parameter gesteuert werden.

2.2.1 Verfahren

Die Symbolisierung durch SAX erfolgt in zwei Schritten: Der Reduktion mittels PAA und der Symbolisierung.

Piecewise Aggregate Approximation

Im ersten Schritt, *Piecewise Aggregate Approximation*, wird die Zeitreihe in Fenster der Länge w eingeteilt. w muss zuvor als Parameter des Algorithmus festgelegt werden und steuert die Kompression der Datenreihe. Anschließend wird über den Werten in jedem der Fenster der Mittelwert bestimmt. Dadurch wird die Zeitreihe C der Länge n komprimiert als Zeitreihe $\overline{C}$ der Länge $\lceil \frac{n}{w} \rceil$ dargestellt. Zur Vereinfachung wird angenommen, dass n durch w ohne Rest teilbar ist. Der i-te Wert $\overline{c_i} \in \overline{C}$ wird definiert durch

$$\overline{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \tag{2.1}$$

Diskretisierung zu SAX-Strings

Um die Werte $\overline{c_i} \in \overline{C}$ zu diskretisieren, werden sie einem Wertebereich v_i zugeordnet. Die Anzahl der Wertebereiche $|V|$ wird durch den Parameter a bestimmt. Anstatt den gesamten Wertebereich $V_{\overline{C}}$ von $\overline{C}$, in a gleichgroße Bereiche einzuteilen, wird $\overline{C}$ in möglichst gleichverteilte Symbole übersetzt. Durch Standardisierung der Zeitreihe $\overline{C}$ kann man von einer Normalverteilung der Werte ausgehen. Die Wertebereiche werden nun so definiert, dass $\frac{1}{a}$ der Werte in jeden Wertebereich fallen.

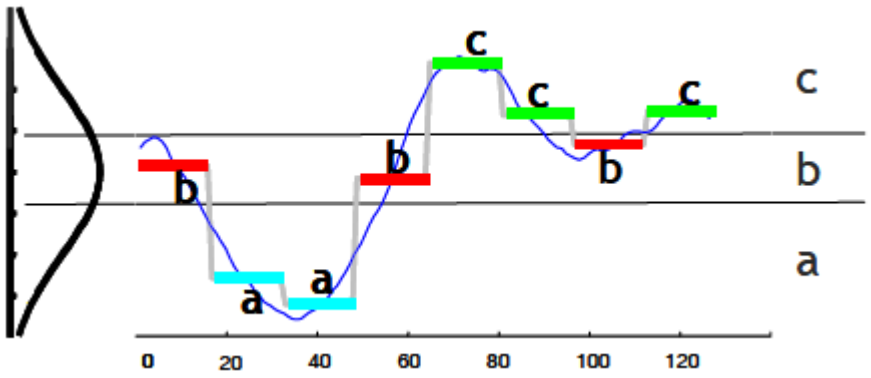


Abbildung 2.1: Einteilung des Wertebereiches in 3 Teilbereiche. Links ist die Normalverteilung eingezeichnet. Die gefensterten Teilstücke der Wertereihe sind, je nach Wertebereich, einem Symbol zugeordnet. *Quelle:* [19]

Die Zuordnung der Werte $\overline{c_i} \in \overline{C}$ in die Wertebereiche, so dass die Anzahl der Werte in jedem Wertebereich gleichverteilt ist, kann über eine Lookuptable realisiert werden. Diese speichert die Grenzen für eine gegebene Anzahl a an Wertebereichen.

Jedem Wertebereich wird nun ein diskretes Symbol zugeordnet und aus der gefensterten, normalisierten Wertereihe $\overline{C}$ wird die diskrete Reihe $\hat{C}$.

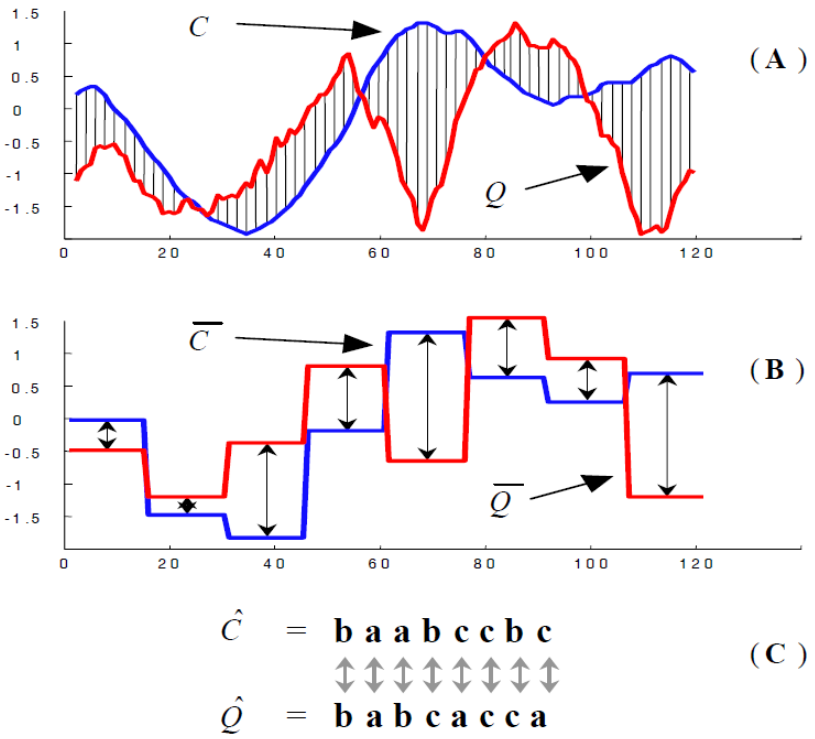


Abbildung 2.2: (A) zeigt zwei original Zeitreihen, (B) deren Partitionierung mit PAA, (C) schliellich die zuordnung zu Symbolen und repräsentation der Wertereihe als SAX-String. Das Distanzmaß zwischen beiden Reihen ist ebenfalls eingezeichnet. *Quelle:* [19]

2.2.2 Distanzmaß

SAX definiert ein Distanzmaß über diskrete Werte indem es die minimale euklidische Distanz $dist(a, b)$ der zu jedem Symbol zugeordneten Wertebereiche berechnet. Sei b_i der Grenzwert, der die Wertebereiche v_i und v_{i+1} trennt, dann ist die minimale euklidische Distanz $dist(a, b)$ definiert als

$$dist(a,b) = \begin{cases} 0, wenn |a-b| \leq 1 \\ b_{max(a,b)-1} - b_{min(a,b)}, sonst \end{cases} \tag{2.2}$$

2.2.3 Einteilung der Werte in Fenster

Zur Vereinfachung wurde bisher angenommen, das n durch w ohne Rest teilbar ist. Dies beschränkt die Wahl des Parameters w auf Teiler von n und fordert implizit Wissen über die Länge der Zeitreihe.

Beispiel

Zur Veranschaulichung des Problems und dessen Lösungen liege folgenden Beispiel vor: Die Wertereihe $C = \{c_1, c_2, \ldots, c_8\}$ bestehe aus 8 Werten und soll in $w = 3$ Fenster eingeteilt werden. Die Länge der Fenster sei somit $l = 2\frac{2}{3}$. Gleichung (2.1) gibt für die Indizes j der Werte in Fenster i folgende Grenzen an:

$$\frac{n}{w}(i-1)+1 \leq j \leq \frac{n}{w}i \tag{2.3}$$

Für Fenster $i = 1$ ergeben sich so alle Indizes $1 \leq j \leq 2\frac{2}{3}$, für Fenster $i = 2$ alle Indizes $2\frac{2}{3} \leq j \leq 5\frac{1}{3}$. Es ist daher unklar, ob c_3 zu Fenster 1 oder Fenster 2 zugewiesen werden sollte.

Eine einfacher Lsungsansatz ist es, nicht ganzzahlige Indizes aufzurunden. Dadurch verkleinert sich das letzte Fenster.

Keogh & Lin schlagen eine Methode vor, Werte anteilig den überschneidenden Fenstern zuzuteilen. Der Wert von c_3 würde so zu $\frac{2}{3}$ zu Fenster 1 und zu $\frac{1}{3}$ zu Fenster 2 addiert werden. Zu beachten gilt es, dass Fenster 1 dann nicht aus 3 sondern aus $2\frac{2}{3}$ Werten besteht und die Berechnung des Mittelwertes dementsprechend angepasst werden muss.

2.3 Symbolisierung durch Clustering

Bei diesem Verfahren wird einem Clusteralgorithmus eine Zeitreihe oder ein Teilstücke einer Zeitreihe als Beispiel mit übergeben. Anhand der Attribute ordnert der Clusterer jedes Beispiel einem Cluster zu. Dabei bilden möglichst ähnliche Beispiele einen Cluster. Je nach Clusteralgorithmus muss die Anzahl der Cluster, durch einen Parameter (meist k), vorgegeben werden. Die Zuordnung der Zeitreihen zu den Clustern dient dabei als diskreter Wert, bzw. Symbol, welche die Zeitreihe von da an repräsentiert.

Die Zeitreihe wird zunächst in Teilstücke eingeteilt, bzw. gefenstert. Die Fenstergröße w wird als Parameter zuvor festgelegt. Lässt sich die Menge der Einzelwerte nicht glatt aufteilen, kann die Größe der Fenster einfach aufgerundet werden, was zu dem Problem führt, dass das letzte Fenster weniger Werte umfasst. Dieses Fenster kann mit Null, dem vorherigen Wert oder mit dem Mittelwert des Fensters aufgefüllt werden, was allerdings zu verfälschten Ergebnissen führen kann. Die beste Lösung ist daher, den Parameter w an die Länge der zu untersuchenden Reihe anzupassen oder eines der Verfahren aus 2.2.3 benutzen.

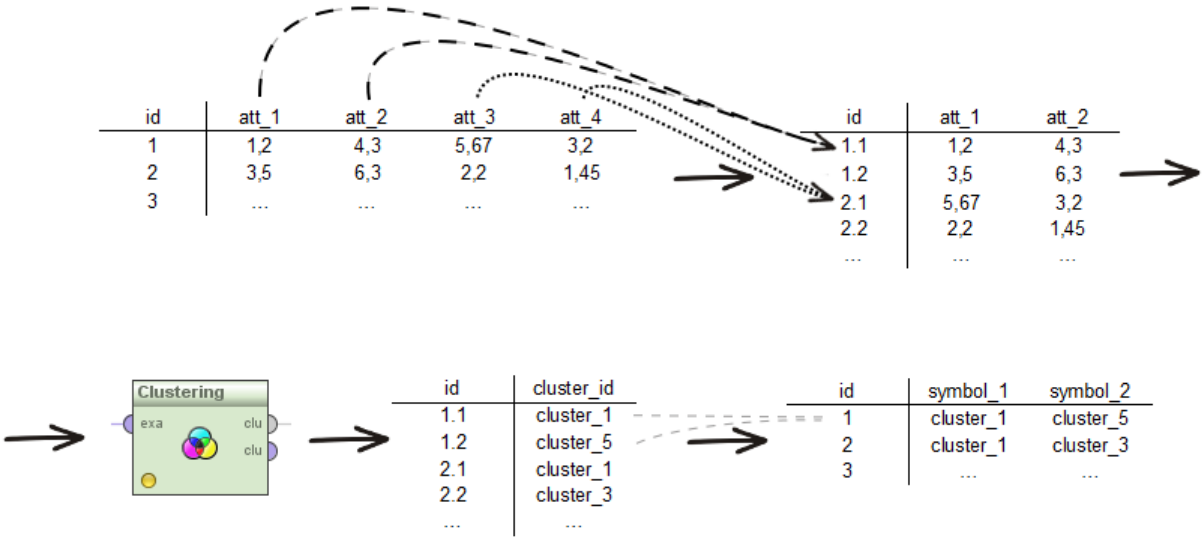


Abbildung 2.3: Schematischer Ablauf der Symbolisierung von Zeitreihen durch Clustering. Jedes Fenster wird zuersts zu einem Beispiel des Examplesets. Das entstandene Exampleset beinhaltet so alle Fenster aller Zeitreihen und wird im folgenden Schritt geclustert. Danach werden alle Zeitreihen aus ihren Fenstern wieder zusammengesetzt, wobei jedes Fenster aus einem Symbol besteht.

Auf der Menge aller Fenster aller Zeitreihen wird nun ein Clustering-Verfahren angewendet. Dabei kann der konkrete Algorithmus aus dem Repertoire der Clustering-Algorithmen des RapidMiner bezogen werden. Es ist denkbar, dass verschiedene Clusteringalgorithmen verschiedene Ergebnisse liefern. So ist es sinnvoll die Auswahl des Cluster-Algorithmus als optimierbaren Parameter offen zu lassen. Jeder Clusterer benötigt verschiedene Parameter, von denen einige zudem, durch den evolutionären Algorithmus, optimiert werden könnten. Geeignete Clustering-Algorithmen sind:

- **k-Means** [14]

  k-Means findet Cluster mit ihren Schwerpunkten in Menge ungelabelter Daten. Die Anzahl der zu identifizierenden Cluster muss zuvor als Parameter k festgelegt worden sein. Nachdem zu jedem Cluster ein intitialer Clusterschwerpunkt gesetzt wurde, werden folgende Schritte so oft wiederholt, bis sich die Clusterschwerpunkte nicht mehr ändern:

  1. jedes Beispiel wird dem Cluster zugeordnet, dessen Clusterschwerpunkt am nächsten liegt.
  2. der Mittelwert über alle Beispiele in einem Cluster, wird als neuer Clusterschwerpunkt gesetzt.

  Die initialen Clusterschwerpunkte werden üblicherweise zufällig gesetzt.

- **Support Vector Machine** [7] [14]

  Die Support Vector Machine (kurz SVM) benötigt gelabelte Daten, für die sie eine Hyperebene berechnet, die als Trennfläche zwischen den Klassen fungiert. Zu klassifizierende Daten können so einer Klasse zugeordnet werden, je nachdem auf welcher Seite der Hyperebene sie liegen. Bei Experimenten mit SVM als Clusteringalgorithmus zu Symbolisierung hat sich jedoch herausgestellt, dass SVM erheblich längere Laufzeit benötigt als andere Algorithmen. So hat die Symbolisierung eines Satzes von Zeitreihen bis zu zwei Stunden in Anspruch genommen. Daher wurde die SVM bei den Experimenten zur Evaluierung der Symbolisierungsverfahren nicht benutzt.

- **DBSCAN** [11]

  DBSCAN benötigt zwei Parameter: ϵ und "minPts". ϵ definiert die maximale Länge, die zwischen zwei Punkten liegen darf, damit diese als "benachbart" gelten. Hat ein Punkt mindestens "mindPts" Nachbarn, so bilden diese ein Kernobjekt. Zudem gelten zwei Punkte als "dichte-verbunden", wenn es eine Kette von Kernobjekten gibt, die diese Punkte miteinander verbinden. So bildet eine Menge von dichteverbundenen Punkten einen Cluster. Punkte die nicht Teil eines Kernobjektes sind, gelten als Rauschen.

Als Distanzmaß zum Clustering der Fenster eignet sich die euklidische Distanz der Attribute jedes Fensters, es ist aber auch denkbar ein schnelleres, approximatives Verfahren anzuwenden [18].

2.4 Weiterverarbeitung der Symbole

Nach der Symbolisierung durch SAX oder Clustering können die Zeitreihen direkt als Eingabe eines Lernverfahrens genutzt werden. Außerdem ist es möglich weitere Verfahren aus dem Bereich des Data-Mining zur Merkmalsextraktion anzuwenden. Die folgende Liste gibt eine Aussicht auf mögliche zusätzliche Merkmale und Methoden:

- einfache statistische Merkmale wie: maximale Distanz, durschnittl. Distanz, Varianz
- Außenseitererkennung [5] [17]
- Mustererkennung [6] [25] [2]
- Mining Frequent Itemsets [3], Associations [1] und Correlations
- Trendanalyse [8]
- Indexing [23]

Kapitel 3

Evolutionäre Merkmalsextraktion

3.1 Evolutionäre Algorithmen (EA) und Genetische Programmierung (GP)

3.1.1 Evolutionäre Algorithmen

Die fenetische Programmierung gehört zur Klasse der evolutionären Algorithmen und ist damit ein Algorithmus zur näherungsweisen Lösung eines Optimierungs- oder Suchverfahrens [22]. Es wird eine Lösung l aus dem Raum R aller Lösungen gesucht, die eine gegebene Zielfunktion möglichst optimal löst.

Ein einfaches Beispiel für die Anwendung der evolutionären Optimierung ist das allgemein bekannte Rucksackproblem. Der Suchraum R des Problems lässt sich als Menge aller Bitstrings der Länge n definieren. Eine Lösung l wäre beispielsweise $(1, 0, 1, 1)$, für ein Rucksackproblem mit 4 Gegenständen. Zielfunktion sei eine Funktion $f(a, b, c, d)$ mit $a, b, c, d \in \{1, 0\}$, welche das Gesamtgewicht und den Wert aller Gegenstände, die die Lösung vorgibt, berechnet. Ist das maximale Gewicht überschritten, gilt die Lösung als ungültig, ansonsten bezeichnet der Wert die Güte der Lösung.

Bei der Suche lehnen sich Evolutionäre Algorithmen an die Grundprinzipien der biologischen Evolution an, was durch die Namensgebung der verschiedener Begriffe deutlich wird.

Zum finden einer optimalen Lösung der Zielfunktion muss zunächst der Suchsraum und eine Fitnessfunktion definiert werden. Die *Fitnessfunktion* dient der Bewertung der Lösungen und ist bei evolutionären Algorithmen meistens die Zielfunktion selbst. Der Suchraum besteht meistens aus allen Möglichen Eingaben der Fitnessfunktion. Eine Menge von Lösungen wird *Population* genannt, jede einzelne Lösung *Individuum*.

Begonnen wird mit einer, meist zufälligen, initialen Population. Ausgehend von dieser wird die Population schrittweise in eine neue Generation überführt, indem die Population zuerst mit der Fitnessfunktion evaluiert wird und dann schlechte Lösungen aussortiert

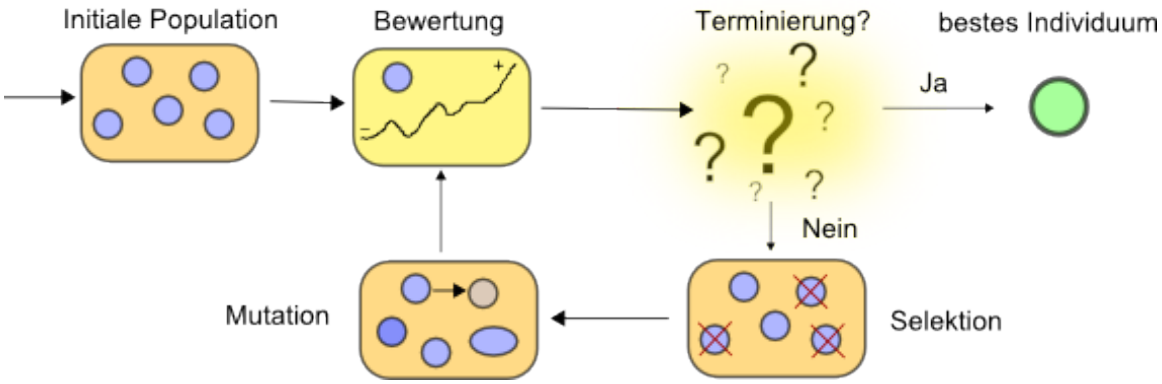


Abbildung 3.1: Schematischer Ablauf eines evolutionären Algorithmus. Begonnen wird mit einer initialen Population. Diese wird durch die Fitnessfunktion bewertet. Treffen Abbruchkriterien zu, so wird Terminiert und das beste gefundene Individuum wird ausgegeben. Ansonsten wird die Population selektiert und mutiert. Nun befindet sich die Population in einer neuen Generation und wird erneut bewertet.

werden. Durch Rekombination zweier Lösungen, Abänderung, zufälliger Generierung und weiteren sogenannten *Mutationen* entsteht schließlich eine neue Generation und die zuvor genannten Schritte werden wiederholt.

Durch diesen Ablauf wird erhofft, immer bessere Lösungen zu finden. Terminiert wird, sobald ein Abbruchkriterium, wie beispielsweise das finden einer perfekten Lösung, erfüllt wurde. Eine Garantie für die Güte der gefundenen Lösungen sowie Laufzeit des Algorithmus gibt es nicht.

Ebenso gibt es keine generellen Richtwerte für die größe der initialen Population, Wahl des Abbruchkriteriums sowie für eine Reihe von weiteren Parametern des Algorithmus wie z.B. maximale Anzahl von Verzweigungen, Tournamen Size, Wahrscheinlichkeit von genetischen Operatoren und maximale Anzahl von Generationen.

3.1.2 Genetische Programmierung

Die Genetische Programmierung ist eine Variante von evolutionären Algorithmen, bei der Individuen als ganze Programme $p(i)$ oder Baumstrukturen ein Programm $p(i)$ beschreiben, interpretiert werden [4]. Der Suchraum R besteht daher aus einem (unendlich großen) Raum von möglichen Programmen $p(i)$, die Input i verarbeiten.

Die Fitnessfunktion $f(p(i))$ bewertet bei der genetischen Programmierung somit nicht eine direkte Lösung, sondern wie gut der Output des Programmes das Problem, für vorgegebenen Input i, löst.

Mutationen der genetischen Programmierung sind meist Operationen auf einer oder mehreren Baumstrukturen.

3.2 Automatisierte Merkmalsextraktion durch Genetischer Programmierung

Komplette Zeitreihen mit mehreren Hunderten Messwerten als direkten Input eines maschinellen Lernverfahrens zu benutzen, ist sehr rechenaufwändig. Weiterhin fehlen in vielen Reihen zeitliche Zusammenhänge: der i-te Messwert einer Wertereihe hat oft keine besondere Bedeutung für das Lernverfahren [20]. Insbesondere trifft dies bei Zeitreihen zu, die Audiodaten in Wellenform repräsentieren. Es gibt nur in seltensten Fällen einen festen Zeitpunkt in einem Lied, dessen Wert Aufschluss über die Eigenschaften des Liedes gibt. Aus diesen Gründen ist es erforderlich, Merkmale aus den Daten zu extrahieren. Da die Menge möglicher Merkmale groß ist und nicht jedes Merkmal bedeutent für ein Lernverfahren ist, soll eine Menge geeigneter Merkmale bestimmt werden.

Da oft kein Wissen über möglichst gute Merkmale von Daten besteht, wird bei [20] versucht, mit einem evolutionären Optimierungsverfahren einen bestmöglichen Vorverarbeitungsprozess zu finden.

3.2.1 Vorverarbeitungsprozesse und Methoden

Vorverarbeitungsprozesse bestehen aus mehreren verschiedenen Methoden. Eine Methode bekommt eine Zeitreihe als Eingabe, führt eine Operation auf ihr aus, und liefert als Ausgabe eine Zeitreihe oder ein oder mehrere Merkmale [20]. Die Aneinanderreihung und/oder Verschachtelung von einer oder mehereren Methoden, bilden einen Vorverarbeitungsprozess. Die Zeitreihe wird dabei von Methode zu Methode weitergereicht. Vorverarbeitungsprozesse können außerdem als Bäume von Methoden angesehen werden. Durch die Baumstruktur eignen sie sich für die Optimierung durch genetische Programmierung.

Die verfügbaren Methoden teilen sich auf in zwei Klassen: Transformationen und Funktionale. Transformationen heißen alle Methoden, die eine (andere) Zeitreihe als Ausgabe liefern [20]. Diese Klasse teilt sich weiterhin auf in Basistransformationen, Filter, Auszeichner und Fensterungen. Auszeichner weisen einem Bereich einer Zeitreihe eine Eigenschaft zu. Basistransformationen bilden die Daten in einen anderen Raum ab, Filter ändern Elemente im gleichen Raum. Fensterungen führen Methoden auf mehreren Teilstücken einer Zeitreihe aus. Abbildung 3.2 zeigt eine Übersicht mit Beispielen. Dabei können sich die Teilstücke überlappen. Funktionale sind Methoden, die als Ausgabe einzelne Zahlwerte liefern. Diese Werte werden der Zeitreihe angehängt und werden als Merkmale bezeichnet.

Weiterhin gibt es Verzweigungen und Ketten. Eine Kette enthält eine oder mehrere innere Methoden die nacheinander ausgeführt werden. Die Ausgabe der letzten inneren Methode ist zugleich Ausgabe der Kette. Eine Verzweigung enthält mehrere Ketten und gibt an jede eine Kopie ihrere eigenen Eingabe weiter. Die Ausgaben aller Ketten werden zusammengeführt, indem alle extrahierten Merkmale an die Zeitreihe der Ausgabe der

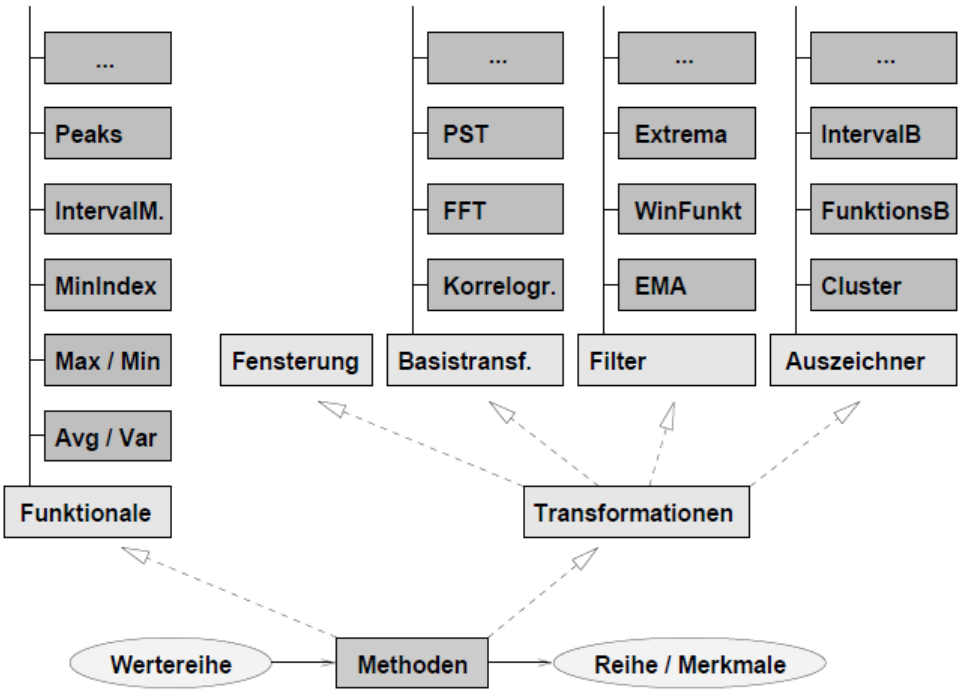


Abbildung 3.2: Übersicht über das System der Vorverarbeitungsmethoden. *Quelle:* [20]

letzten Kette angefügt werden. Optional können auch alle Zeitreihen aller Ketten behalten werden.

SAX und Symbolisierung durch Clustering wurden jeweils als Transformation und Funktional integriert. Als Funktional fügen die Methoden jedes Symbol als Merkmal hinzu. Ein Merkmal besteht aus einem Namen und einem numerischen Wert. Wird eine Zeitreihe nach der Symbolisierung mit SAX durch n Symbole dargestellt, so erhält die Zeitreihe n Merkmale mit den Namen SAX_1, SAX_2, ..., SAX_n. Anstatt diskrete Werte als Merkmalswerte zu nutzen, bietet es sich an, jedes Symbol durch einen numerischen Wert darzustellen, wodurch Distanzen zwischen Symbolen einfach berechnet werden können. Als festen Wert für ein Symbol eignet sich bei SAX der durch PAA ermittelte Mittelwert eines Fensters, beim Clustering der Centroid des Fensters aus dem gelernten Clustermodel. Diese Werte werden auch bei der Symbolisierung als Transformation genutzt. Die Zeitreihe wird durch eine neue Zeitreihe mit n Werten ersetzt.

3.2.2 Fitnessfunktion

Zur Evaluierung der Individuen werden die zugehörigen Vorverarbeitungsprozesse auf einer Menge von gelabelten Zeitreihen angewandt. Jeder so entstandene Datensatz wird als Eingabe für ein Lernverfahren genutzt. Der konkrete Lernalgorithmus muss zu Beginn der automatischen Merkmalsextraktion vorgegeben worden sein. Für Experimente mit größeren Datenmengen bietet sich eine Kreuzvalidierung zur Messung der Performanz an. Dafür

wird auf einem Teil der Daten ein Modell gelernt, welches dann auf dem restlichen Teil der Daten angewandt wird. k mal wiederholt erhält man über das Mittel aller Performanzen eine k-fache Kreuzvalidierung.

Die Performanz des Lernverfahrens wird zum Schluss ausgewertet und als Fitnesswert dem entsprechenden Individuum zugeordnet. Somit entspricht die Fitnessfunktion des evolutionären Algorithmus der Performanz des Lernalgorithmus auf den vorverarbeiteten Daten und ist somit ebenfalls von diesem abhängig. Daher sollten verschiedene Lernalgorithmen ausprobiert werden, um eine qualitativere Aussage über die Güte der automatischen Merkmalsextraktion machen zu können.

3.2.3 Mutationsoperatoren und Selektionsstrategien

Nach Ausführung der Vorverarbeitungsprozesse und Evaluiering durch die Fitnessfunktion wird die Population durch Selektion verkleinert und Mutationsoperatoren verändern Prozesse.

Selektionsstrategien sind Verfahren zur Auswahl von Individuen, die in die nächste Generation übergehen. Nicht ausgewählte Individuen werden gelöscht. Durch das ValueSeries-Plugin stehen folgende Selektionsverfahren zur verfügung:

- **Tournament**

  Der Tournament-Algorithmus wählt zufällig zwei Individuen aus der Population und. Das Individum mit der höheren Fitness wird zur neuen Population hinzugefügt. Die Menge der Individuen die zum Tournament “antreten“ ist durch den Parameter *tournamentsize* wählbar.

- **Roulette Wheel Selection**

  Roulette Wheel Selection, auch Fitnessproportionale Selektion genannt, teilt jedem Individuum eine Wahrscheinlichkeit zu, mit der es gewählt wird. Die Wahrscheinlichkeit ist dabei proportional zur Fitness des Individuums. Das Verfahren kann als Roulettetisch veranschaulicht werden, bei dem jedem Individuum ein Fach des Roulettetisches zugewiesen wird. Das Individuum wird gewählt, wenn die (zufällig rollende) Roulettekugel in dem Fach landet. Die Größe des Faches ist proportional zur Fitness des Individuums. Eine hohe Fitness erhöht somit die Wahrscheinlichkeit gewählt zu werden. Die Anzahl der auszuwählenden Individuen ist als Parameter wählbar.

Bei jeder Selektionsstrategie ist es zudem möglich ,dass das Individuum mit der höchsten Fitness in der Population direkt ausgewählt wird (*elitist selection*).

Nachdem die Individuen der neuen Generation ausgewählt wurden, werden genetische Operatoren (Mutationsoperatoren) auf einigen Individuen angewandt. Genetische Operatoren dienen bei evolutionären Algorithmen zur Mutation der genetischen Merkmale der

Individuem. Bei der genetischen Programmierung werden dementstrechend die Methodenbäume der Individuen verändert. Das ValueSeriesplugin bietet drei Mutationen an:

- **Generierungs-Mutation**

  Die Generierungs-Mutation fügt an einer zufälligen Stelle eines Methodenbaumes eine zufällige Methode ein.

- **Löschungs-Mutation**

  Die Löschungs-Mutation löscht eine zufällig gewählte Methode aus dem Methodenbaum.

- **Wechsel-Mutation**

  Die Wechsel-Mutation welchselt eine Methode aus dem Methodenbaum gegen eine zufällig gewählte andere Methode aus.

- **Parameter-Mutation**

  Diese Mutation ändert Parameter einer Methode aus dem Methodenbaum. Welche Parameter auf welche Weise geänderte werden, geben die Methoden selbst in einer öffentlichen Methode vor.

- **Crossover**

  Crossover kreuzt zwei Individuen, indem es die genetischen Merkmale beider Individuen kombiniert. Bei den Methodenbäumen der genetischen Programmierung wird die Kombination durch Austauschen von Teilbäumen realisiert.

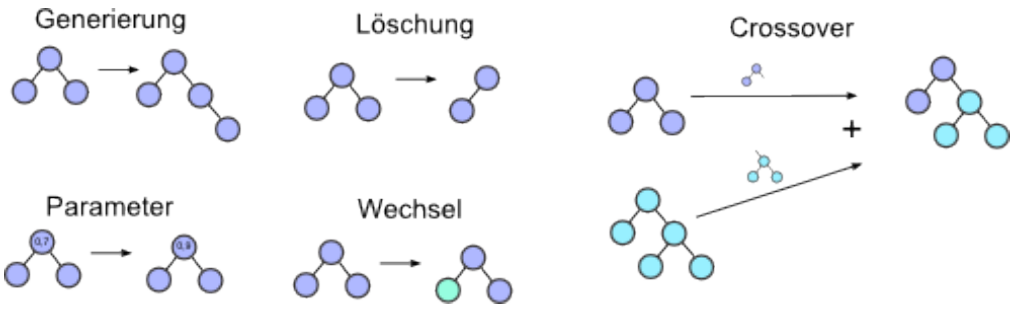


Abbildung 3.3: Mutationen der genetischen Programmierung.

3.2.4 Ablauf des GP-Algorithmus

Der Ablauf des GP-Algorithmus basiert hauptsächlichauf einer simplen Schleife, in der alle Individuen bewertet werden, die Population durch Selektion mit einem der zuvor vorgestellten Selektionsverfahren in eine neue Generation übergeht und einige Individuen schließlich mutiert werden. Eine Skizze dieser Schleife ist in Algorithmus 1 gelistet.

Algorithmus 1 Ablauf des evolutionären Algorithmus

Eingabe: Gelabelte Zeitreihen

Ausgabe: Vorverarbeitungsprozess

```
bildet initiale Population
while Abbruchkriterium erfüllt? do
    bewerte Individuen
    Selektiere Individuen
    wende genetische Operatoren an (Mutationen)
end while
```

Kapitel 4

Integration der Symbolisierungsverfahren

4.1 RapidMiner

RapidMiner ist eine open source Experimentierumgebung für maschinelles Lernen und Data-Mining, welche 2001 vom Lehrstuhl für künstliche Intelligenz der TU Dortmund, unter dem Namen YALE, entwickelt wurde. Durch das ValueSeries Plugin erhält RapidMiner eine Datenstruktur und verschiedene Methoden zur Zeitreihenanalyse. Bis zur Version 4.6 war das zuvor bereits erwähnte Verfahren zur automatischen Merkmalsextraktion aus Audiodaten Bestandteil des ValueSeries Plugins. Aktuell steht RapidMiner in Version 5.3 zur Verfügung. Daher war es das Ziel die automatische Merkmalsextraktion an die Neuerungen des RapidMiner 5.3 anzupassen, um die Symbolisierungsmethoden aus 2 zu erweitern und Experimente mit Daten der Fallstudie des Stahlwalzwerkes durchzuführen.

4.1.1 Einleitung in die Mechanismen des RapidMiner

Ein Datenverarbeitungsverfahren in RapidMiner wird *Prozess* genannt. Ein Prozess beinhaltet einen oder mehrere *Operatoren*. Operatoren stellen je eine Methode zur Datenverarbeitug, Datengenerierung oder Datenanalye dar. Einige Operatoren sind in der Lage selbst einen Prozess zu beinhalten, welcher Subprozess genannt wird. Ihren Input bekommen Operatoren durch einen oder mehrere Inputports, die Ausgabe geht an einen oder mehrere Outputports. So können Operatoren Daten von anderen Operatoren annehmen, verarbeiten, und wiederum an andere Operatoren weiterleiten. Die wichtigste Klasse zur Datenhaltung stellt dabei das *ExampleSet* dar. Ein ExampleSet ist eine Tabelle, deren Zeilen Beispiele und Spalten Attribute verkörpern. Um einen gültigen Prozess zu erstellen und starten müssen alle benötigten Operatoren in einen Prozess gesetzt, der Datenverlauf durch Verbindungen von Ports definiert und optional Parameter der Operatoren gesetzt

werden. Eine GUI vereinfacht die Erstellung von Prozessen, welche in der Auszeichnungsprache *XML* definiert werden, wesentlich.

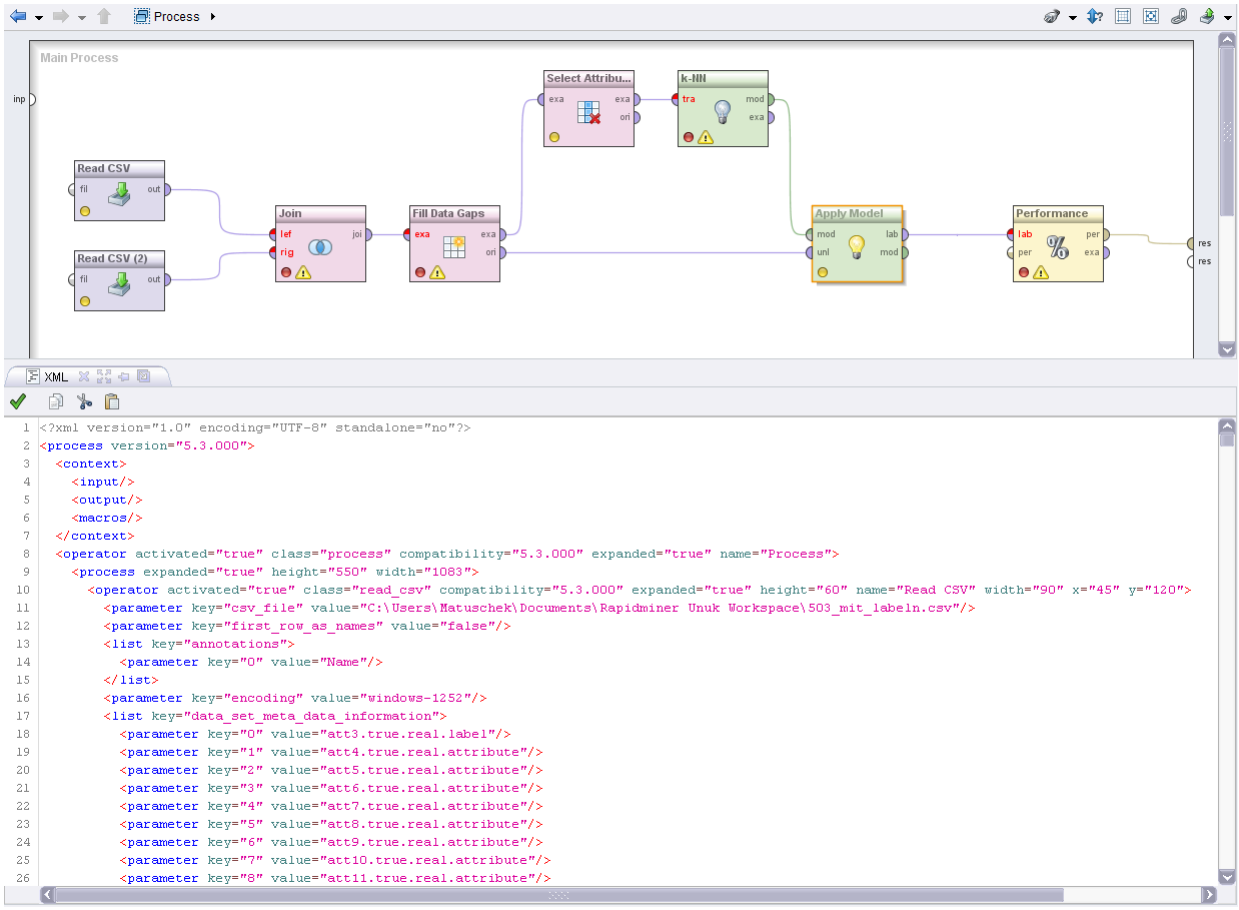


Abbildung 4.1: Beispiel eines gültigen RapidMiner-Prozesses in der Darstellung durch die GUI (oben) und als XML (unten). Daten werden durch spezielle Import-Operatoren (“Read CSV“) im .csv-Format von der Festplatte geladen und als ExampleSet-Objekt an weitere Operatoren übergeben. Verbindungen von Ports werden durch Linien zwischen den jeweiligen Ports visualisiert. Der Operator “k-NN“ stellt eine Nächste-Nachbarn-Klassifikation dar, welche ein Modell berechnet und weitergibt (grüne Linie).

4.2 Das ValueSeries Plugin

Das ValueSeries Plugin wurde für YALE entwickelt und bis zur aktuellen Version 5.3 fortlaufend aktualisiert. Zentrale Klasse des Plugins ist die Klasse *ValueSeries*, welche eine besondere Datenstruktur für Zeitreihen darstellt. Die meisten Operatoren der Erweiterung sind auf Operationen mit ValueSeries ausgelegt, und nicht auf andere Datenstrukturen anwendbar.

Ab Version 5.3 fehlt der Operator *ValueSeriesGP*, welcher die automatischen Merkmalsextraktion durch einen *genetic programming* Algorithmus implementiert. Daher war das

Ziel der Vorarbeiten dieser Bachelorarbeit, den ValueSeriesGP-Operator an die neueste Version des RapidMiner anzupassen.

Um einen Eindruck der Funktionsweise des ValueSeries-GP Operators zu bekommen, wurde daher versucht, eine automatische Merkmalsextraktion in RapidMiner 4.6 (bzw. 4.5) zu starten, was jedoch nicht gelang. Deshalb wurde beschlossen, den ValueSeriesGP Operator in die neuste RapidMiner Version 5.3 zu integrieren.

Die größte Änderung von RapidMiner 4.6 zu 5.3 ist die Einführung der Ports. Zuvor wurden Prozesse in einer einfachen Baumansicht dargestellt. Der Verlauf der Daten ergab sich durch die Reihenfolge der Operatoren im Baum. In RapidMiner 5.3 können Operatoren auf einem zweidimensionalen Feld frei plaziert werden und der Datenverlauf wird durch das Verbinden von Ports definiert. Bei der automatischen Erstellung und Modifizierung (siehe 3.2.3, Seite 15) von Vorverarbeitungsprozessen musste daher die automatische und korrekte Verbindung der Ports der Operatoren implementiert werden. Wichtig dabei war, nach der Ausführung eines Vorverarbeitungsprozesses, alle Portverbindungen wieder zu lösen. Dies ist nötig, da sich die Anordnung der Operatoren nach einer Mutation aïdern, und so den Datenverlauf stören kann.

Weiterhin hat sich der Name der Methode, welche die Ausführung eines Operator startet, geändert. Die Aufgabe der überholten Methode ***public synchronized final*** *IOContainer apply(IOContainer input)* wurde von ***public void*** *doWork()* übernommen. Im Rahmen dieser Bachlorarbeit war es jedoch nicht möglich alle Fehler in der Ausführung der automatischen Merkmalsextraktion zu korrigieren. Daher kommt es teils zum Wurf von Ausnahmen, welche die direkte Löschung eines Individuums zur Folge haben. In wie weit dies die automatische Merkmalsextraktion durch evolutionäre Optimierung beeinflusst, ist unklar.

4.2.1 Erweiterte Ausnahmebehandelung

Bei der zufallsbasierten Erzeugung von Vorverarbeitungsprozessen kommt es vor, dass Prozesse nicht auf die gegebenen Daten anwendbar sind. Werden z.B. mehrere Fensterungen geschachtelt, unterschreitet die Fenstergröße irgendwann die minimale Fenstergröße. Desweiteren sind viele weitere Transformationen im ValueSeries-Plugin enthalten, welche eine bestimmte Länge der Zeitreihe vorraussetzen.

Bei der Ausführung eines solchen ungültigen Vorverarbeitungsprozesses kommt es daher zwangsläufig zum Wurf einer Java-Ausnahme. Da diese Ausnahmen zu erwarten sind, werden sie vom GP-Algorithmus abgefangen, der Vorverarbeitungsprozess wird gelöscht und die Ausnahme geloggt.

Bei der Erweiterung des Algorithmus möchte der Entwickler jedoch diese Probleme, die durch genetische Veränderungen der Operatoren und deren Parameter entstehen, von

sonstigen Programmierfehlern unterscheiden und diese evtl. nicht loggen, um einen Überblick über Fehler der Operatoren zu bekommen.

Daher wurde eine neue Ausnahmeklasse erstellt, *GPOperatorException*, und der Wurf dieser an allen Stellen im Quellcode eingefügt, an denen solche geplante Fehler entstehen können.

4.3 Der Genetic Programming Operator

Der GP-Algorithmus zur automatischen Merkmalsextraktion befindet sich zum größten Teil in dem Operator *ValueSeriesGP*. Dieser beinhaltet die Schleife des GP-Algorithmus aus 1 und verwaltet Eingabedaten, Population von Vorverarbeitungsprozessen, Evaluationverfahren (Fitnessfunktion), Selektionalgorithmen und Mutationsoperatoren.

Obwohl der Operator zur automatischen Merkmalsextraktion speziell für Zeitreihen implementiert wurde, erwartet der GP-Operator ein ExampleSet als Eingabe. Jedes Beispiel in dem ExampleSet stellt eine Zeitreihe dar, jedes Attribut einen Messpunkt der Reihe. Da die Anzahl von Attributen eines ExampleSets fest ist, ist die Länge jeder Reihe gleich. In ExampleSets besteht die Möglichkeit sogenannte *Missing Values*, also *keinen Wert*, in das Attributfeld zu setzen. ValueSeries fehlt dies jedoch, so dass alle Reihen gleichlang sein müssen, oder kürzere Reihen mit Nullen aufgefüllt werden. Das Auffüllen mit Nullwerten wird jedoch von Zeitreihen-Operatoren nicht berücksichtigt, und führt so zu verfälschten Ergebnissen bei Methoden wie beispielsweise dem Mittelwert über die gesamte Reihe. Als Lösung des Problems wurden alle Zeitreihen durch Interpolierung auf die selbe Länge gebracht. Da aber auch dieses Vorgehen zur Verfälschung der Eigenschaften der Reihen führt, liegt die spätere Anpassung des Plugins an unterschiedlich Lange Zeitreihen nahe. Zur Vorverarbeitung werden alle Zeitreihen aus dem ExampleSet in ein ValueSeries-Objekt und für die Evaluierung wieder in ein ExampleSet überführt. Dies ist notwendig, da Lernalgorithmen des RapidMiner nur für ExampleSet-Objekte implementiert wurden.

Die Population der evolutionären Algorithmen wird vom ValueSeries-GP-Operator als Menge von Individuen verwaltet. Jedes Individuum besitzt eine Operatorkette, in der die eigentlichen Operatoren zur Vorverarbeitung liegen, und einen Fitnesswert.

Der ValueSeriesGP-Operator ist ebenfalls als Operatorkette implementiert. Dem inneren Prozess werden die vorverarbeiteten Daten als ExampleSet zugeführt, und ein PerformanceVector-Objekt wird als Rückgabe erwartet. Dieser PerformanceVector bestimmt die Performanz des Lernverfahrens auf den vorverarbeiteten Daten und dient als Fitnesswert des Individuums.

Die Selektionsverfahren des evolutionären Algorithmus sind als RapidMiner-Operatoren implementiert und liegen dem ValueSeriesGP Operator vor. Welche Methode benutzt wird, muss als Parameter des ValueSeriesGP Operators eingestellt werden. Der Benutzer hat da-

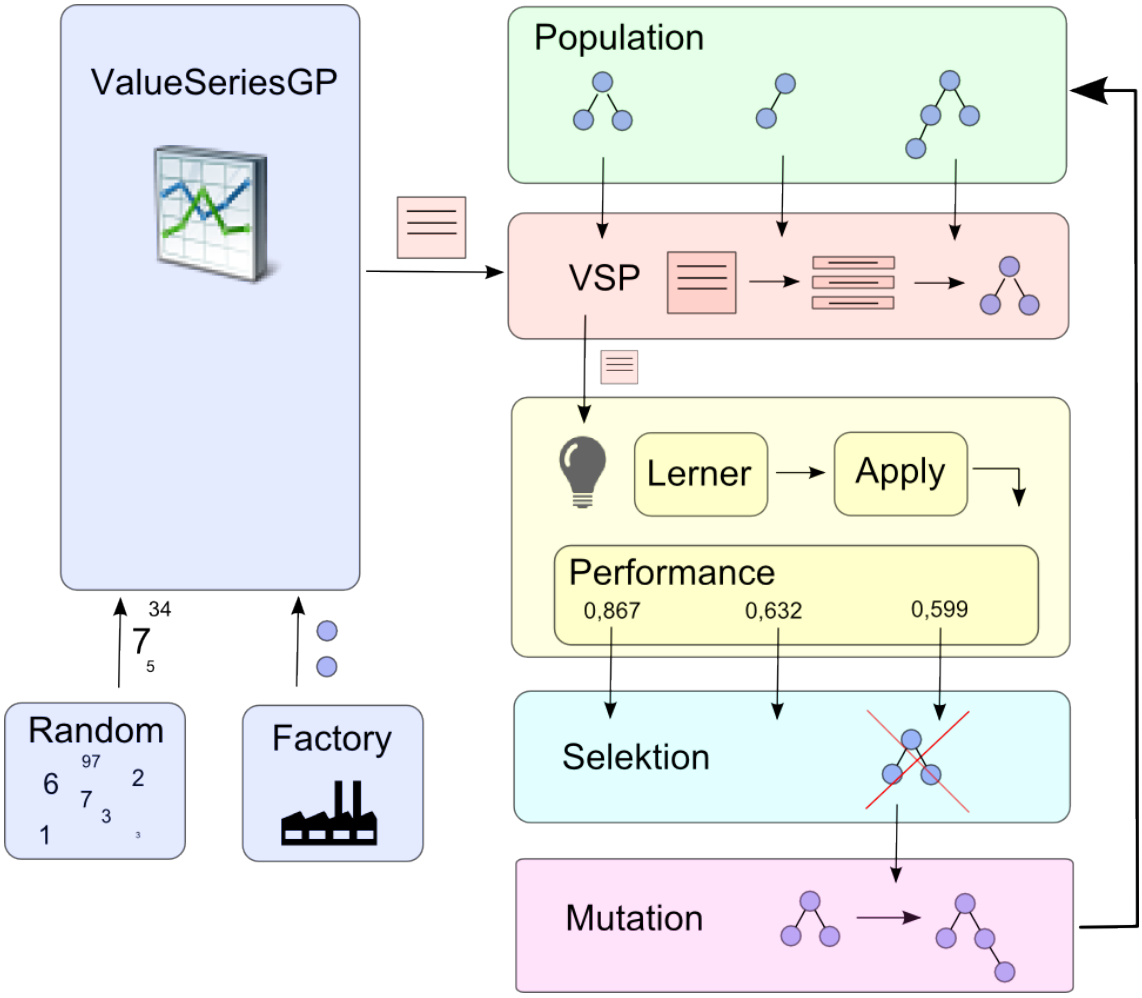


Abbildung 4.2: Skizzierung der automatischen Merkmalsextraktion. ValueSeriesGP erzeugt eine Population mit Hilfe eines Zufallsgenerator und der Operator-Factory, welche RapidMiner-Operatoren erzeugt. Durch einen ValueSeriesPreprocessing-Operator (VSP) je Individuum werden die Zeitreihen einzeln auf den Vorverarbeitungsprozess angewandt. Wieder zu einer Tabelle zusammengefügt werden die vorverarbeiteten Zeitreihen einem Lernverfahren (Lerner) übergeben. Das gelernte Modell wird auf den selben Daten angewandt und ein Performance-Vektor zu jedem Vorverarbeitungsprozess wird erstellt. Dieser ist Basis für die anschließende Selektion. Nach der Mutation befindet sich die Population in einer neuen Generation und der Ablauf beginnt von vorne, bis ein Abbruchkriterium erfüllt wurde.

bei die Wahl zwischen *Tournament Selection* und *Roulette Wheel.*(siehe 3.2.3). Ebenso hält der ValueSeriesGP Operator alle Mutationsmethoden als Operatoren.

4.3.1 Implementierung der neuen Methoden

Um SAX und die Symbolisierung durch Clustering in die evolutionäre Optimierung zu integrieren, wurden beide Verfahren als *RapidMinerValueSeries*-Operatoren implementiert. Um als Funktion zu agieren, und dem ValueSeries-Objekt die Symbole als Merkmale

hinzuzufügen erben die Klassen *ExtractSAX* und *ExtractSubseriesClustering* von der abstrakten Klasse *Function*. Als Transformation fungieren die Klassen *SAXTransformation* und *SubseriesClusteringTransformation*.

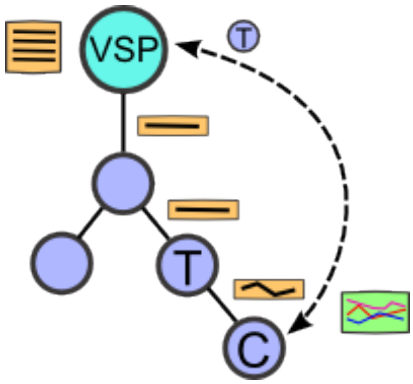


Abbildung 4.3: Die Abbildung zeigt einen Vorverarbeitungsprozess des ValueSeries Plugins als Baumstruktur. Der Knoten mit Beschriftung "T" repräsentiert einen Transformationsoperator, der Knoten mit Beschriftung "C" einen Operator, der die Symbolisierung durch Clustering implementiert. Die Klasse *ValueSeriesProprocessing* (VSP) steht am Anfang jedes Vorverarbeitungsprozesses und ist für die Extrahierung einzelner Zeitreihen aus der Datenhaltung als ExampleSet zuständig. Bei Ausführung des Symbolisierungsoperators, übergibt dieser alle bisher ausgeführten Transformationen an den VSP Operator. Der VSP Operator erstellt ein Clustermodel und gibt es zurück an den Symbolisierungsoperator.

Bei der Integration der Symbolisierung durch Clustering kam es zu Schwierigkeiten, welche jedoch überwunden werden konnten. Fenster auf einer Zeitreihe sollen einem Cluster zugeordnet werden, die Zuordnung soll dieses Fenster als Symbol repräsentieren. Um ein Fenster von Werten einem Cluster zuordnen zu können wird ein Clustermodel benötigt. Um dieses zu berechnen benötigt man wiederum möglichst alle Fenster der Reihe.

Dem Vorverarbeitungsprozess wird jedoch stets nur eine ValueSeriesobjekt, d.h. eine Zeitreihe, zur Vorverarbeitung ausgehändigt. Daher ist es nicht möglich in der Vorverarbeitung jeder Zeitreihe, die Clusterzuordnung durchzuführen. Deshalb wurde die Berechnung des Clustermodels in die Klasse *ValueSeriesPreprocessing* ausgelagert. Diese Klasse hat die Aufgabe das ExampleSet, welches zur Vorverarbeitung von der Klasse *ValueSeries-GP* ausgehändigt wird, zu einer Menge von Zeitreihen als ValueSeries umzuwandeln und dem Vorverarbeitungsprozess zu übergeben. Das Clusteringverfahren wird vom Clustering-Operator angestoßen, um eine unnötige Ausführung, wenn kein Clustering-Operator im Vorverarbeitungsprozess existiert, zu ersparen.

Problematisch wird dieses Verfahren, wenn die Zeitreihe auf dem Weg zum Clustering-Operator transformiert wird. Durch die Transformation sind die Zeitreihen, mit denen das Clustermodel berechnet wurde, und die Zeitreihen die schließlich darauf angewendet werden, von einander verschieden. Eine sinnvolle Zuordnung ist nicht mehr möglich. Daher sucht der Clustering-Operator, vor Anstoßen der Berechnung des Clustermodels,

nach Transformationen, die die Zeitreihen bisher passiert haben. Kopien dieser Operatoren werden dem ValueSeriesPreprocessing Operator übermittelt. Bevor dieser nun das Clustermodel berechnet, wendet er die Transformationen auf alle Zeitreihen an. Da die Transformations-Operatoren des ValueSeries-Plugins nur auf ValueSeries-Objekte anwendbar sind, der ValueSeriesPreprocessing-Operator aber alle Zeitreihen in einem ExampleSet hält, ist es erforderlich, alle Zeitreihen zuerst aus dem ExampleSet zu extrahieren, auf die Transformationen anzuwenden, und wieder als ExampleSet zusammenzufassen. Abbildung 4.3 illustriert die Struktur des beschriebenen Verfahrens.

Nach der Symbolisierung durch SAX und Clustering ist es denkbar Methoden aus anderen Bereichen des maschienellen Lernes auf den Daten anzuwenden. Um diese in den Prozess der evolutionären Optimierung einzubinden, müssen die Methoden als RapidMiner-Operatoren, die als Eingabe ein ValueSeries-Objekt erwarten, vorliegen. Da die meisten Methoden des RapidMiners ExampleSets erwarten, ist deren Anwendung nicht direkt möglich. Ebenfalls ist es problematisch, auf Symbolreihen spezialisierte Methoden als Operatoren dem ValueSeries-Plugin hinzuzufügen, da es keinen Mechanismus gibt, der sicher stellt, dass zuvor eine Symbolisierung stattgefunden hat. Befindet sich solch ein Operator im Prozessbaum vor der Symbolisierung oder in einem anderen Zweig, könnte er nicht angewandt werden. Als Zwischenlösung wurden einfache Methoden wie das *häufigst vorkommende Symbol* direkt in die Symbolisierungsoperatoren integriert.

Kapitel 5

Evaluation der Symbolisierungsverfahren

5.1 Ziele der Experimente

Um den Einfluss der integrierten Symbolisierungsverfahren auf die automatische Merkmalsextraktion von Zeitreihen einschätzen zu können, werden mehrere Experimente durchgeführt. Es werden Experimente zur automatischen Merkmalsextraktion ohne die in dieser Arbeit vorgestellten Symbolisierungsverfahren angewandt, um vergleichen zu können, wie diese im Vergleich zu Experimenten mit Symbolisierungsverfahren abschneiden. Weiterhin werden unterschiedliche Klassifizierer benutzt, um die Ergebnisse weniger vom angewandten Klassifizierungsalgorithmus abhängig zu machen.

Alle Experimente werden in der Software RapidMiner 5.3 mit ValueSeries Plugin ausgeführt. Als Testdaten dienen sechs verschiedene Sensordatenreihen aus einem Stahlwalzkraftwerk. Dabei handelt es sich um Angaben über Temperatur der Stahlblöcke, Drehzahl und Anstellung der Walzen sowie die aufgebrachte Walzkraft. Da der GP-Algorithmus nur für univariante Zeitreihen konstruiert wurde, müssen alle Datensätze in einem eigenem Experiment untersucht werden. Zusammenhänge zwischen den Daten fließen daher nicht mit in die Auswertung der Vorverarbeitungsprozesse und die Lernverfahren ein.

Bei der Verarbeitung der Zeitreihen durch die Vorverarbeitungsprozesse aus [20] kann es zu Problemen kommen, da diese Zeitreihen gleicher Länge vorraussetzt. Daher wurden die Daten durch Interpolierung auf gleiche Länge gebracht.

Weiterhin wurden die Daten im Vorhinein so behandelt, dass einzelne Stiche (ein Durchlauf des Materials durch die Walze) differenziert untersucht werden können. Dafür wurden die Anzahl der Messdaten innerhalb eines Stichs auf eine feste Anzahl gebracht. Stiche mit abweichender Menge von Messdaten wurden interpoliert um sie dann skaliert wieder als Messreihe abzubilden. Indem der Standardwert für Fensterungen auf der Zeitreihe auf den eben diesen Wert gesetzt wurde, entspricht eine Fensterung der Wertereihe einer

Untersuchung der einzelnen Stiche. Dabei wird gehofft, dass Merkmale aus einzelnen Stichen eines Walzvorgangs, bedeutende Informationen für das Lernverfahren beinhalten. Als fester Wert wurden 100 Messpunkte pro Stich gewählt.

5.2 Versuchsaufbau

Der Grundaufbau eines Experimentes besteht aus dem Laden der Daten, Evaluierung der Merkmalsextraktion und Sicherung der Ergebnisse. Zum Laden der Daten und Sichern der Ergebnisse wurden die RapidMiner-Operatoren *Retrieve* und *Write Performance* benutzt. Der Operator *Validation* stellt eine Kreuzvalidierung dar.

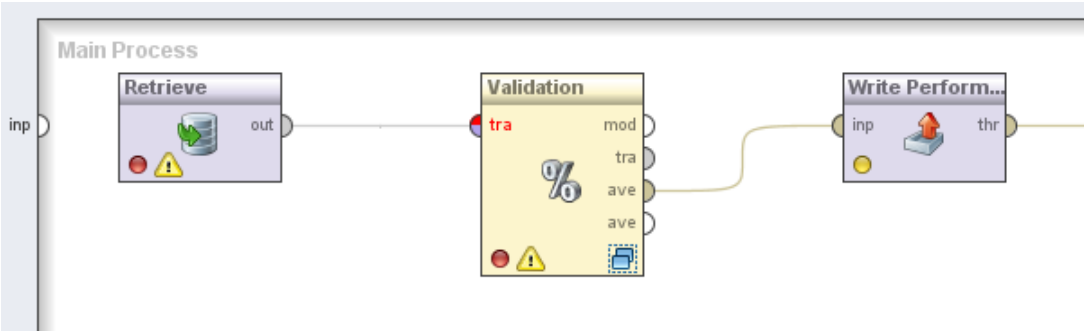
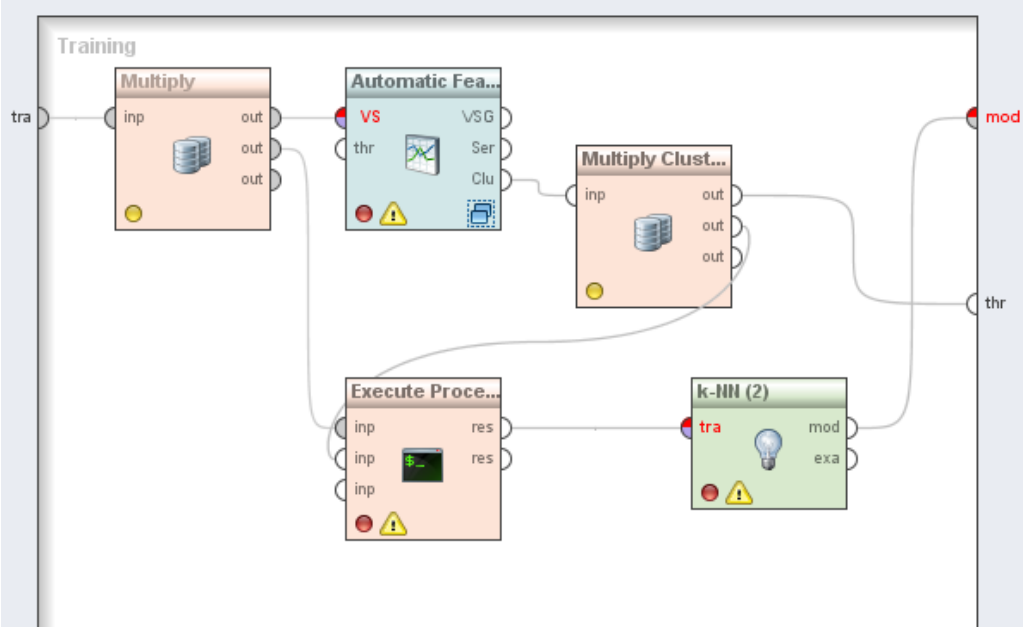


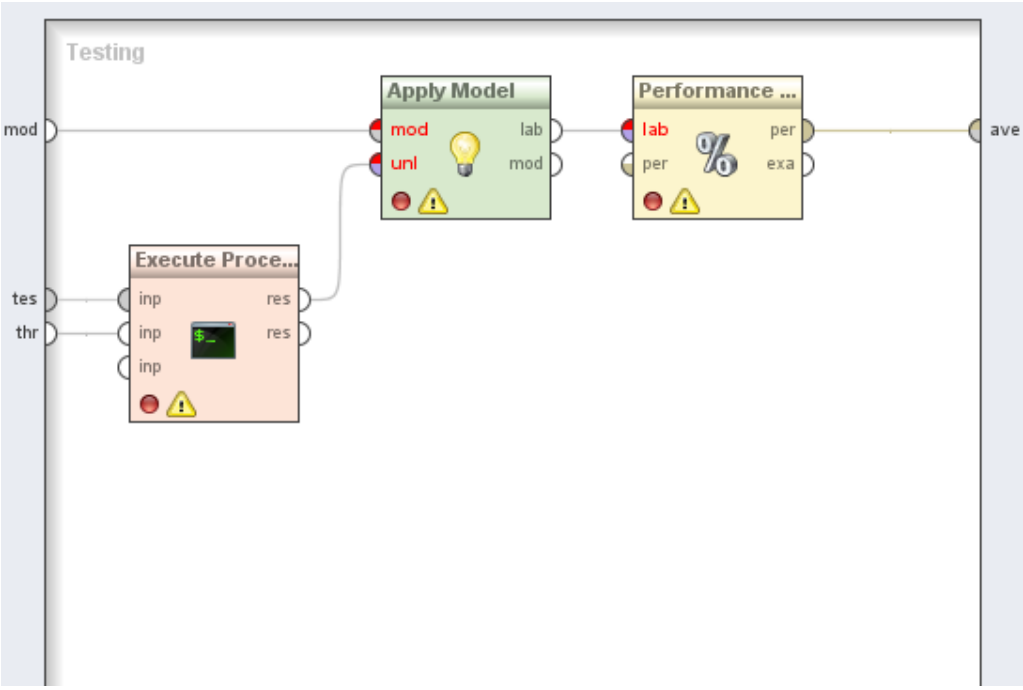
Abbildung 5.1: Hauptprozess des Grundaufbaues der Experimente. *Retrieve* lädt Daten und gibt sie als ExampleSet an die Kreuzvalidierung (*Validation*) weiter. *Write Performance* speichert die Auswertung der Kreuzvalidierung.

Innerhalb der Kreuzvalidierung befinden sich zwei Subprozesse: *Training* und *Testing*. Im Trainingsprozess wird die automatische Merkmalsextraktion auf einem Teil der Daten ausgeführt und in Testing das Ergebniss auf dem Rest der Daten angewandt. Durch 10-maliges Aufteilen der Daten in Test- und Trainingsmenge wird eine qualitative Aussage über die Performanz der automatischen Merkmalsextraktion erwartet.

Im Trainingsprozess wird die Menge an Trainingsdaten durch einen *Multipy Operator* dupliziert. Ein Duplikat wird der automatischen Merkmalsextraktion zugeführt. Die sucht nach einem optimalen Vorverarbeitungsprozess hinsichtlich Klassifikation. Der konkrete Klassifikationsalgorithmus wird im Subprozess des ValueSeriesGP Operators eingefügt. Nach Terminierung dieses Operators muss ein Klassifikationsmodell mit den Vorverarbeiteten Daten erstellt werden und dem Testing-Prozess übergeben werden. Da es im RapidMiner keine Möglichkeit zur Weitergabe von Prozessen gibt, speichert ValueSeriesGP den gefundenen Vorverarbeitungsprozess als XML-Datei auf die Festplatte des Systems. Der Operator *Execute Process* ermöglicht die Ausführung eines gespeicherten Prozesses. Mit dem Duplikat der Trainingsdaten führt Execute Process den Vorverarbeitungsprozess aus, gibt die Daten an eine zusätzliche Instanz des Klassifizierungsoperators und reicht das gelernte Klassifizierungsmodell durch den Port *mod* weiter an den Testprozess.



(a) Subprozess "Training" der Kreuzvalidierung mit dem ValueSeriesGP-Operator (hier: *Automatic Feature Extraction*).



(b) Subprozess "Testing" der Kreuzvalidierung.

Abbildung 5.2: Subprozesse der Kreuzvalidierung

Für den Fall, dass die Symbolisierung durch Clustering Teil des Vorverarbeitungsprozesses ist, gibt der ValueSeriesGP Operator das dort gelernte Clustermodel aus. Dieses

wird dupliziert (*Multiply Cluster*) und an die zusätzlichen Ausführungen des Vorverarbeitungsprozesses (*Execute Prozess*) gereicht.

Im Trainingsprozess werden die Trainingsdaten auf dem, vom ValueSeriesGP gefundenen, Vorverarbeitungsprozess und auf das Klassifizierungsmodell aus dem Trainingsprozess angewandt. Anschließend bewertet der Operator *Performance (Classification)* die Klassifizierung. Der Validationoperator gibt schließlich ein *PerformanceVector* Objekt aus, welches die Performanz des Testprozesses, also der Vorverarbeitung und der Klassifikation, bewertet.

Für die Bewertung der Vorverarbeitungsprozesse in der evolutionären Optimierung des ValueSeriesGP Operators wurde der Prozess aus Abbilung 5.3 erstellt. Es fällt auf, das nicht wie in Abschnitt 3.2.2 (Seite 14) empfohlen, eine Kreuzvalidierung gewählt wurde. Der Grund dafür ist, das die bereits eine 10-fache Kreuzvalidierung um die automatische Merkmalsextraktion und anschließender Klassifizierung gesetzt wurde. Die angewandten Testdaten umfassen je Datensatz 500 Zeitreihen. Durch 10-fache Kreuzvalidierung bekommt der ValueSeriesGP Operator nur $\frac{1}{10}$, also 50 Zeitreihen. Eine weitere 10-fache Kreuzvalidierung würde die bereits sehr niedrige Anzahl von Zeitreihen zur Beurteilung von Vorverarbeitungsprozessen auf 5 Zeitreihen senken. So kann es vorkommen, dass der Klassifizierer alle 5 Zeitreihen zufällig richtig klassifiziert. Dies entspricht einer Genauigkeit vonn 100%, also einer perfekten Klassifikation. Der evolutionäre Algorithmus terminiert und gibt den vermeintlich perfekten Vorverarbeitungsprozess aus. Auf einem größeren Datensatz wird diese triviale Klassifizierung ("alles gehört zu einer Klasse") allerdings zu brauchbaren Ergebnissen führen. Daher wird auf dem gesamten Satz von Zeitreihen, die dem ValueSeriesGP Operator zur verfügung stehen, ein Modell gelernt und evaluiert.

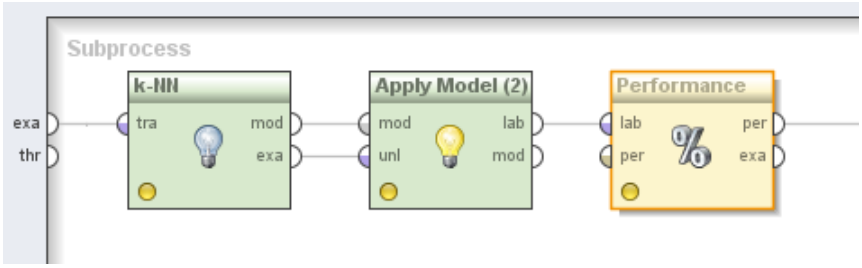


Abbildung 5.3: Subprozess zur Evaluierung der Vorverarbeitungsprozesse. Dient der evolutionären Optimierung der automatischen Merkmalsextraktion als Fitnessfunktion.

5.2.1 Klassifizierer

Um die Bewertung der automatischen Merkmalsextraktion weniger unabhängig vom verwendeten Klassifizierungsalgorithmus zu machen, werden jeweils vier Experimente mit vier verschiedenen Klassifizierern durchgeführt. Zusätzlich wurden Parametereinstellun-

gen der Klassifizierungsalgorithmen durch Ausprobieren optimiert. Im folgenden werden die verwendeten Klassifizierer gelistet und kurz vorgestellt:

- **k-NN**

  Der k-NN- oder *k-Nächste-Nachbarn*-Algorithmus entscheidet, zu welcher Klasse ein Objekt gehört, indem die k Objekte analysiert werden, die im Raum am nächsten zum dem Objekt liegen. Das Objekt wird der Klasse zugewiesen, welche unter den k nächsten Nachbarn am häuftigen vertreten ist. k ist dabei ein frei wählbarer Paramter.

- **Support Vector Machine**

  Die Support Vector Machine für Klassifikation arbeitet ähnlich wie die SVM zum Clustering (siehe Kapitel 2.3).

- **Naive Bayes**

  Die Naive Bayes Klassifikation entscheidet, zu welcher Klasse ein Objekt gehört, indem es die Wahrscheinlichkeiten vergleicht, mit denen ein Objekt einer Klasse angehören könnte. Die Klasse, zu der das Objekt am wahrscheinlichsten gehört, wird dem Objekt schließlich zugeordnet.

- **Descision Tree**

  Bei der Klassifikation per Entscheidungsbaum, liegt ein Baum vor, dessen Blätter Klassen repräsentieren. Jede Abzweigung beschreibt Werte eines Attributes. Um ein Objekt zu Klasifizieren wird bei dem Wurzelknoten angefangen und jeweils die Abzweigung gewählt, die bei dem Objekt zutrift. So gelangt man zwangsläufig zu einer Klasse im Blattknoten, die dem Objekt zugeordnet wird.

5.2.2 Konfigurationen

Es wurden drei Parameter zur Konfiguration der Experimente gewählt.

- 6 Datensätze
- 4 Klassifizierer
- 2 Operatoren-Setups

Jeder Datensatz enthält Meßdaten eines anderen Sensors der Stahlwalze. Diese wurden mit allen vier Klassifizierern, *Naive Bayes*, *Support Vector Machine*, *k-NN* und *Decision Tree*, kombiniert. Zuletzt wurde jedes Experiment mit den neu integrierten Symbolisierungsverfahren und ohne durchgeführt. Dadurch kommt man auf 48 verschiedene Experimente.

Andere Parameter, wie etwa maximale Anzahl an Generationen der evolutionären Optimierung oder Parameter der Klassifizierer wurden zuvor manuell optimiert. Die folgende Liste beschreibt wichtige Parameter dieser Grundeinstellung:

- **Kreuzvalidierung** Anzahl der Validierungen: *10*
- **k-NN** k : *10*
- **Decision Tree** maximale Tiefe : *10*
- **Support Vector Machine** C : *200*
- **SAX**
  - Anzahl der Fenster: *5*
  - Größe des Alphabetes: *14*
- **Symbolisierung durch Clustering**
  - Fensterlänge: *100*
  - Anzahl der Cluster: *5*
- **ValueSeriesGP**
  - höchste Generation: *80*
  - maximale Anzahl neuer Operatoren: *3*
  - Wahrscheinlichkeit der Parametermutation: *0,3*
  - Wahrscheinlichkeit der Generierungsmutation: *0,2*
  - Wahrscheinlichkeit der Löschungsmutation: *0,2*
  - Wahrscheinlichkeit der Wechselmutation: *0,2*
  - Wahrscheinlichkeit der Crossovermutation: *0,4*

5.3 Ergebnisse

Bei der Durchführung der Experimente, die eine Support Vector Machine zur Klassifikation benutzen, konnten keine Ergebnisse berechnet werden. Grund dafür ist die, im Gegensatz zu anderen Klassifizierern, hohe Laufzeit der SVM. Die Laufzeit eines der Experimente mit SVM Klassifizierer ließ sich auf ca. 60 Tage schätzen.

Die Ergebnisse der Experimente sind in Abbildung 5.4a und 5.4b graphisch dargestellt. Es lassen sich deutliche Unterschiede zwischen den Klassifizierern feststellen. Naive Bayes schneidet im Schnitt sehr schlecht ab und erzielt teilweise nur Klassifikationsgenauigkeiten von ca. 65%. Die Klassifikationsgenauigkeit misst den Anteil an Beispielen einer Testmenge, die richtig klassifiziert worden sind [13].

Bei den Angaben über Decision Tree fällt auf, dass bei jedem Datensatz der selbe Wert von 85,42% erzielt wurde. Dieser Wert entsteht, indem der Klassifizierer alle Beispiele der Klasse zuordnet, zu der am meisten Beispiele vorhanden sind. In diesem Fall wurden alle 487 Zeitreihen als "Gut" klassifiziert, wobei 416 der Beispiele wirklich "Gut" sind und 71 falsche Negative darstellen. Der Wert für die Klassifikationsgenauigkeit und der Vorhersagegenauigkeit der Klasse "Gut" beträgt somit

$$\frac{\text{Beispiele der Klasse "Gut"}}{\text{Alle Beispiele}} = \frac{416}{487} = 85,42\% \tag{5.1}$$

k-NN erzielt konstant gute Ergebnisse und erzielt auch die höchste Genauigkeit von 86,24%.

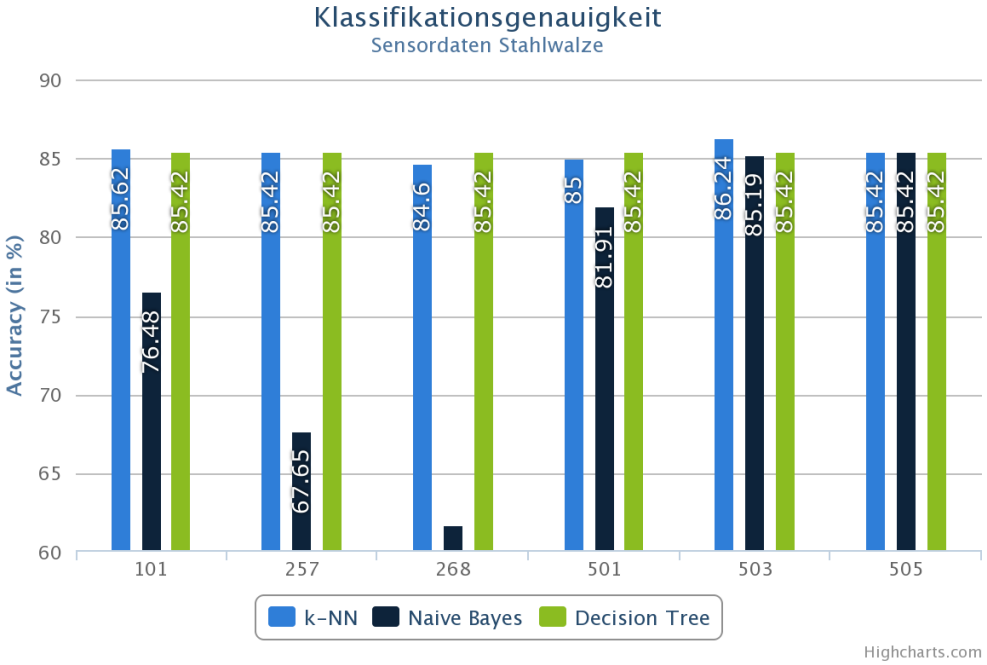
Die Ergebnisse der Experimente ohne Symbolisierungsverfahren weichen nur geringfügig von denen der Experimente mit Symbolisierung ab. Hier ist k-NN ebenfalls der beste Klassifizierer, Decision Tree erzielt bei jedem Experiment 85,42% und Naive Bayes schwankt zwischen sehr schlechten und guten Klassifikationsgenauigkeiten. Der Wert von 86,24% durch k-NN mit Symbolisierung kann jedoch nicht erreicht werden.

5.4 Auswertung

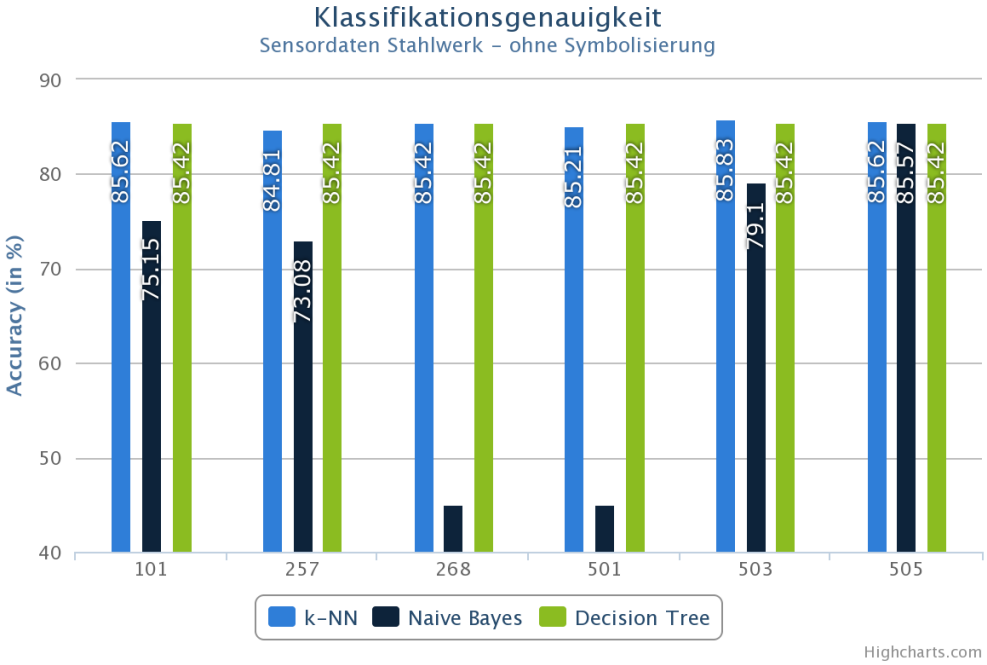
Die bisher erzielte Vorhersagegenauigkeit von 80,21% in bisherigen Experimenten konnte durch die automatische Merkmalsextraktion übertroffen werden. Obwohl der Klassifizierer k-NN durchgehend hohe Werte erzielt, steigt die Klassifikationsgenauigkeit nie deutlich höher als 85,42% und erzeugt daher keine, für die Qualitätsprognose, brauchbaren Werte.

Bei der genaueren Untersuchung der gefundenen Vorverarbeitungsprozessen fällt jedoch auf, dass die integrierten Symbolisierungsmethoden häufig vorkommen. Inbesondere *Sax Extraction* ist Bestandteil in 12 von 18 Vorverarbeitungprozessen, die durch die evolutionäre Optimierung gefunden worden sind. Von den übrigen sechs Vorverarbeitungsprozesse benutzt einer *Cluster Extraction* und ein weiterer *SAX Transformation*. Damit enthalten 14 von 18 gefundenen Vorverarbeitungsprozessen eine der Symbolisierungsmethoden. Im Schnitt umfassen die gefundenen Vorverarbeitungsprozesse 12 Methoden, wovon keine ähnlich oft vorkommt, wie *SAX Extraction*. Dadurch lässt sich nicht auf eine gute Performanz der Vorverarbeitung mit *SAX Extraction* schließen, deutet dies jedoch an.

Eine deutliche Verbesserung durch die Integration der Symbolisierungsverfahren ist nicht nachweisbar, da beide Experimentkonfigurationen sehr ähnliche Klassifikationsgenauigkeiten erzielt haben. Jedoch ist diese Evaluation stark von den benutzten Daten abhängig. Es ist möglich, dass die Sensordaten der Stahlwalze keinen Schluss über die Qualität des Endproduktes zulassen, was eine hinreichend genaue Klassifizierung unmöglich macht.



(a) Klassifikationsgenauigkeiten nach automatischer Merkmalsextraktion.



(b) Klassifikationsgenauigkeiten nach automatischen Merkmalsextraktion ohne Symbolisierung.

Abbildung 5.4: Plots der Ergebnisse der Experimente zur Klassifikation mit automatischer Merkmalsextraktion mit Sensordaten einer Stahlwalze. Die Zahlen *102* bis *505* bezeichnen den benutzten Datensatz und dessen Herkunft.

Kapitel 6

Zusammenfassung und Aussicht

In den Vorarbeiten zu dieser Arbeit wurde der RapidMiner-Operator zur automatischen Merkmalsextraktion aus dem ValueSeries Plugin an die neuste RapidMiner Version 5.3 angepasst. Geringfügige Programmierfehler bestehen jedoch weiterhin.

Dem ValueSeries Plugin wurden vier Operatoren zu zwei verschiedenen Symbolisierungsverfahren von Zeitreihen hinzugefügt, welche von der automatischen Merkmalsextraktion genutzt werden können. Die Implementierung weiterer Verfahren zur Vorverarbeitung auf Basis der symbolischen Darstellung von Zeitreihen konnte nicht integriert werden, da dies eine erhebliche Umstrukturierung des ValueSeriesGP Operators vorraussetzt.

Zur Evaluierung der um Symbolisierung erweiterten Merkmalsextraktion wurden Experimente mit Sensordaten aus einem Stahlwalzwerk ausgeführt. Die Lernaufgabe bestand darin, die Qualität des Endproduktes durch Klassifikation als “Gut“ oder “Schlecht“ einzustufen. Kein gefundener Vorverarbeitungsprozess konnte die für diese Experimente benutzen Daten so aufbereiten, dass ein Klassifizierungsalgorithmus ein hinreichend genaues Modell erstellen konnte. Um eine Verbesserung der automatischen Merkmalsextraktion durch Symbolisierung zu verdeutlichen, wurde jedes Experiment mit und ohne Symbolisierungsverfahren, als Option der evolutionären Optimierung, ausgeführt. Dies gelang jedoch nicht, da die Ergebnisse beider Konfigurationen ähnlich ausfielen.

Grund für die ungenügend genaue Klassifikation könnten die Sensordaten selbst sein. Es ist möglich, dass die Daten keine Merkmale enthalten, die für die Qualitätsprognose relevant sind. Desweiteren handelt es sich mit 500 Zeitreihen pro Sensor-Datensatz um relativ wenig Beispiele. Mit größeren Datensätzen wäre auch der Einsatz einer Kreuzvalidierung im Evaluationsprozess des ValueSeriesGP-Operators denkbar, wodurch die Bewertung der Fitness genauer ausfallen könnte.

Weiterhin wären mehrere Erweiterungen der automatischen Merkmalsextraktion des ValueSeries Plugins denkbar. Durch Anpassungen des Operators könnten die Vorverarbeitungsprozesse Zeitreihen mit unterschiedlichen Längen verarbeiten, wodurch eine In-

terpolierung der Daten überflüssig wäre. So könnten die Daten unverfälscht bearbeitet werden.

Die Verarbeitung von multivarianten Zeitreihen könnte, durch evolutinonär optimierte Vorverarbeitungprozesse, Zusammenhänge finden, die für die anschließenden Lernalgorithmen von Bedeutung sind.

Methoden aus anderen Bereichen des Data-Mining auf den Symbolisierten Daten anzuwenden würde die Möglichkeiten der Vorverarbeitung von Zeitreihen erheblich erweitern. Bisher ist die automatische Merkmalsextraktion beschränkt auf Operatoren zur Verarbeitung von ValueSeries Objekten.

Für diese Erweiterungen sind jedoch erhebliche Umstrukturierungen und Anpassungen der automatischen Merkmalsextraktion und der Operatoren des ValueSeries Plugins nötig.

Anhang A

Hinzugefügte und geänderte Java-Klassen

Für das Upgrade des ValueSeriesPlugin für Rapidminer 4.6 auf Rapidminer 5.3 mussten mehrere Java-Klassen geändert oder neue Klassen angelegt werden. Die Folgende Liste zeigt alle veränderten Java-Klassen des RapidMiner ValueSeries-Plugins. Neue Klassen werden mit einem + gekennzeichnet.

Klassenname	Beschreibung u. vollständiger Klassenname
NewOperatorFactory	com.rapidminer.valueseries.NewOperatorFactory Erzeugt RapidMiner-Operatoren für die Vorverarbeitungsprozesse.
ExtractSAX	com.rapidminer.valueseries.functions.ExtratSAX Implementiert den in Kapitel 2 vorgestellten SAX-Algorithmus. Die diskreten Werte werden hier als Feature der Zeitreihe hinzugefügt, ebenso wie darauf aufbauenden Merkmale aus Kapitel 2.4.
+ SAXTransformation	com.rapidminer.valueseries.transformations.SAXTransformation Implementiert den in Kapitel 2 vorgestellten SAX-Algorithmus. Die Zeitreihe wird hier durch die Mittelwerte der zu den Symbolen zugeordneten Wertebereiche ersetzt.
+ AutomaticOperatorChain	com.rapidminer.valueseries.AutomaticOperatorChain Erweitert die OperatorChain um automatische Portverbindung der Operatoren im Subprozess.
ValueSeriesPreprocessing	com.rapidminer.valueseries.ValueSeriesPreprocessing Überführt ein ExampleSet in ein ValueSeries Objekt, wendet einen Vorverarbeitungsprozess auf diesen an und speichert alle Wertereihen wieder in ein ExampleSet.
ExtractSubseriesCluster	com.rapidminer.valueseries.functions.ExtractSubseriesCluster Implementiert den vorgestellten Algorithmus zum fensterbasierten Zeitreihenclustering. Das Ergebnis wird in Form von Attributen gespeichert.
+ GPOperatorException	com.rapidminer.valueseries.gp.GPOperatorException Identifiert einen automatisch erstellten Vorverarbeitungsprozess als nicht ausführbar. Beispielsweise, wenn die letzten 10 Werte einer Wertereihe abgeschnitten werden sollen, aber die Wertereihe selber weniger als 10 Werte beinhaltet.
ValueSeriesGP	com.rapidminer.valueseries.gp.ValueSeriesGP Zentrale Klasse für die Ausführung des GP-Algorithmus zur automatischen Merkmalsextraktion von Wertereihen.
+ SubseriesClusterTransformation	com.rapidminer.valueseries.transformations Implementiert den vorgestellten Algorithmus zur Symbolisierung durch Clustering. Die Werte der Zeitreihe werden durch Mittelwerte der gefundenen Clusterzentroide ersetzt.
BranchingOperator	com.rapidminer.valueseries.gp.BranchingOperator Realisiert eine Verzweigung des Datenverarbeitungsprozesses. Dieser Operator wurde an den RapidMiner 5.3 angepasst.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Agrawal, Rakesh, Tomasz Imieliński Arun Swami: *Mining association rules between sets of items in large databases. ACM SIGMOD Record*, 22, 207–216. ACM, 1993.

[2] Agrawal, Rakesh Ramakrishnan Srikant: *Mining sequential patterns. Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, 3–14. IEEE, 1995.

[3] Agrawal, Rakesh, Ramakrishnan Srikant .: *Fast algorithms for mining association rules. Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 1215, 487–499, 1994.

[4] Banzhaf, Wolfgang, Peter Nordin, Robert E. Keller Frank D. Francone: *Genetic Programming : An Introduction : On the Automatic Evolution of Computer Programs and Its Applications (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann Publishers, 1997.

[5] Barnett, Vic Toby Lewis: *Outliers in statistical data*, 1994.

[6] Bishop, Christopher M Nasser M Nasrabadi: *Pattern recognition and machine learning*, 1. springer New York, 2006.

[7] Burges, Christopher J. C.: *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Min. Knowl. Discov., 2(2):121–167, 1998.

[8] Chatfield, Chris: *The analysis of time series: an introduction*. CRC press, 2003.

[9] Daw, C. S., C. E A Finney E. R. Tracy: *A review of symbolic analysis of experimental data*. Review of Scientific Instruments, 74(2):915–930, 2003.

[10] Dimitrova, Elena S., Paola Vera-Licona, John McGee Reinhard C. Laubenbacher: *Discretization of Time Series Data*. Journal of Computational Biology, 17(6):853–868, 2010.

[11] Ester, Martin, Hans peter Kriegel, Jrg S Xiaowei Xu: *A density-based algorithm for discovering clusters in large spatial databases with noise.* 226–231. AAAI Press, 1996.

[12] Gupta, Ankit, Kishan G. Mehrotra Chilukuri Mohan: *A clustering-based discretization for supervised learning.* Statistics Probability Letters, 80(910):816 – 824, 2010.

[13] Han, J. M. Kamber: *Data Mining: Concepts and Techniques.* Morgan Kauffman, 2001.

[14] Hastie, Trevor, Robert Tibshirani, Jerome Friedman James Franklin: *The elements of statistical learning: data mining, inference and prediction.* The Mathematical Intelligencer, 27(2):83–85, 2005.

[15] Jain, Anil K. Richard C. Dubes: *Algorithms for clustering data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[16] Jin, Xiaoming, Jianmin Wang Jiaguang Sun: *Dynamic Symbolization of Streaming Time Series.* Yang, ZhengRong, Hujun Yin RichardM. Everson (): *Intelligent Data Engineering and Automated Learning IDEAL 2004*, 3177 *Lecture Notes in Computer Science*, 559–564. Springer Berlin Heidelberg, 2004.

[17] Knox, Edwin M Raymond T Ng: *Algorithms for mining distance-based outliers in large datasets. Proceedings of the International Conference on Very Large Data Bases.* Citeseer, 1998.

[18] Kriegel, Hans-Peter, Peer Kröger, Alexey Pryakhin, Matthias Renz Andrew Zherdin: *Approximate Clustering of Time Series Using Compact Model-Based Descriptions.* Haritsa, Jayant R., Kotagiri Ramamohanarao Vikram Pudi (): *DASFAA*, 4947 *Lecture Notes in Computer Science*, 364–379. Springer, 2008.

[19] Lin, Jessica, Li Wei et al.: *Experiencing SAX: a Novel Symbolic Representation of Time Series*, 2007.

[20] Mierswa, Ingo: *{A}utomatisierte {M}erkmalsextraktion aus {A}udiodaten*, 2004.

[21] Mörchen, Fabian Alfred Ultsch: *Optimizing time series discretization for knowledge discovery. Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, 660–665, New York, NY, USA, 2005. ACM.

[22] Nissen, V.: *Einführung in Evolutionäre Algorithmen.: Optimierung nach dem Vorbild der Evolution.* Computational intelligence. [Wiesbaden : Vieweg]. Vieweg Friedr. + Sohn Ver, 1997.

[23] O'Neil, Patrick Dallan Quass: *Improved query performance with variant indexes. ACM Sigmod Record*, 26, 38–49. ACM, 1997.

[24] Sant'Anna, A. N. Wickstrom: *Symbolization of time-series: An evaluation of SAX, Persist, and ACA. Image and Signal Processing (CISP), 2011 4th International Congress on*, 4, 2223–2228, 2011.

[25] Waterman, Michael S .: *Introduction to computational biology: maps, sequences and genomes.* Chapman & Hall Ltd, 1995.

[26] Zhou, Feng, F. Torre J.K. Hodgins: *Aligned Cluster Analysis for temporal segmentation of human motion. Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on*, 1–7, 2008.