Exposé für die Bachelorarbeit: Approximation des k-NN Klassifikators im ClaSP-Algorithmus

Betreuer: Prof. Dr. Ulf Leser, Arik Ermshaus

Institution: Humboldt Universität zu Berlin

Benjamin Waibel

November 2022

1 Einleitung

Menschen können in vielen Bereichen wiederkehrende Ereignisse oder Muster erkennen. Für Analysen komplexer, unübersichtlicher Datensätze, beispielsweise in Form von Aktienkursen oder EKGs, ist dies jedoch nicht trivial. Ein Teil dieser Analysen besteht aus dem Erkennen unterschiedlicher Muster. So könnte dies bei einem Aktienkurs ein Auf- oder Abwärtstrend sein [17]. Des Weiteren können EKGs Aufschluss darüber geben, ob eine zu untersuchende Person zu einer bestimmten Zeit unter epileptischen Anfällen [10] oder Panikattacken [5] leidet.

Für Computer ist die Erkennung solcher Phasen keine triviale Aufgabe. Sie können jedoch dazu beitragen, Teilbereiche in den Datensätzen zu identifizieren, um auf Veränderungen im zugrunde liegenden Verhalten zu schließen [17, 10, 5]. Dazu gibt es im Forschungsbereich der Zeitreihenanalyse Methoden, Segmente von Zeitreihen ausfindig zu machen. Das beschriebene Problem findet insbesondere eine Lösung in der Zeitreihensegmentierung. Zeitreihen sind Messdaten, die über einen fortlaufenden Zeitraum mit einer konstanten Abtastrate aufgezeichnet werden. Die Zeitreihensegmentierung hilft dabei, (semantisch) unterschiedliche Bereiche einer Zeitreihe ausfindig zu machen [11]. Diese unterschiedlichen Bereiche werden Segmente genannt.

2 Zielsetzung

Für das Problem der Zeitreihensegmentierung gibt es in der Informatik viele Lösungen [8]. Akkurate Lösungen sind unter anderen BinSeg [13], FLOSS [6] und ClaSP [11]. Letzterer wird in dieser Arbeit genauer betrachtet. Hierbei besteht das Ziel, Zeitreihen mithilfe von bestehenden Approximationsverfahren zu diskretisieren, um dadurch den k-NN Klassifikatior im ClaSP-Algorithmus zu verschnellern. Genauer gesagt wird untersucht, wie sich die gegenwärtige Python-Implementierung von ClaSP [4] in Bezug zu einer angepassten Implementierung mit SAX [7] bzw. SFA [12] auf Laufzeit und Genauigkeit auswirkt. Die Approximationsverfahren SAX und SFA transformieren eine Zeitreihe in eine symbolische und kürzere Sequenz in Form eines Strings. Die Motivation für diese Arbeit liegt in der Verringerung der Laufzeit des ClaSP-Algorithmus, die für sehr große Datensätze hoch ist. Dies ist wichtig, da Zeitreihensegmentierung ein typischer Vorverarbeitungsschritt für lange Zeitreihen ist [11].

ClaSP ist in verschiedene Komponenten unterteilt. Dazu gehören die Aufteilung der Zeitreihe in gleich große überlappende Fenster und die Berechnung der k nächsten Nachbarn (kNN) als Preprocessing für die Kreuzvalidierung (engl. Cross Validation). Darüber hinaus ist die anschließende Änderungspunktbestimmung und Segmentierung Teil von ClaSP [11].

Das Ziel dieser Arbeit ist es, die bisherige Implementierung zur Ermittlung der kNN anstelle des STOMP-Algorithmus [18] in ClaSP zu ersetzen. Zu diesem Zweck werden die beiden Approximationsverfahren SAX und SFA verwendet, die zur Berechnung

der kNN beitragen sollen. Dabei sollen die Laufzeit und die Genauigkeit der beiden Implementierungen analysiert und diskutiert werden, wobei eine geringere Laufzeit bei möglichst geringem Genauigkeitsverlust von ClaSP im Vergleich zur bisherigen Implementierung in [4] angestrebt wird.

3 Grundlagen und verwandte Arbeiten

In diesem Abschnitt werden die Grundlagen behandelt. Dazu gehören die Definition der Zeitreihe, der Teilsequenz einer Zeitreihe, der Segmentierung einer Zeitreihe sowie die des k-NN Klassifikators. Darüber hinaus wird die Vorgehensweise von ClaSP, SAX und SFA im Einzelnen betrachtet.

Definition 1. Eine Zeitreihe ist eine Folge von $n \in \mathbb{N}$ reellen Werten $T = (t_1, ..., t_n), t_i \in \mathbb{R}$ [11]. So sind, wie eingangs beschrieben, beispielsweise Elektrokardiogramme (EKG) oder Aktienkurse Zeitreihen [12]. Dabei betrachten wir lediglich synchronisierte Zeitreihen, bei denen die Dauer zwischen t_i und t_{i+1} stets konstant ist.

Definition 2. Die Teilsequenz $T_{s,e}$ einer Zeitreihe T mit Startpunkt s und Endpunkt e enthält die aufeinander folgenden Werte von T von Position s zu e, d. h. $T_{s,e} = (t_s, ..., t_e)$ mit $1 \leq s \leq e \leq n$. Die Länge von $T_{s,e}$ beträgt $|T_{s,e}| = e - s + 1$ [11].

Definition 3. Eine Segmentierung einer Zeitreihe T in $C + 1$ Segmente ist eine geordnete Folge von Wechselpunkten $t_{i_1}, ..., t_{i_C}$ mit $1 < i_1 < ... < i_C < n$ [11].

Definition 4. Das Problem der Zeitreihensegmentierung (engl. time series segmentation (TSS)) besteht darin, eine sinnvolle Segmentierung einer gegebenen Zeitreihe T unter der Annahme zu finden, dass T durch einen Prozess mit diskreten Zuständen erzeugt wurde. Eine Segmentierung wird als sinnvoll erachtet, wenn die Wechselpunkte zwischen zwei aufeinander folgenden Segmenten Zustandsänderungen im zugrunde liegenden Prozess entsprechen. [11]

Definition 5. Ein k-NN Klassifikator klassifiziert ein gegebenes Sample, indem er die k nächstgelegenen Samples in den Trainingsdaten unter Verwendung einer vordefinierten Distanzfunktion findet. Anschließend bestimmt er das vorhergesagte Label, indem er das Majoritätslabel der k-NN Trainingsproben ausgibt [11].

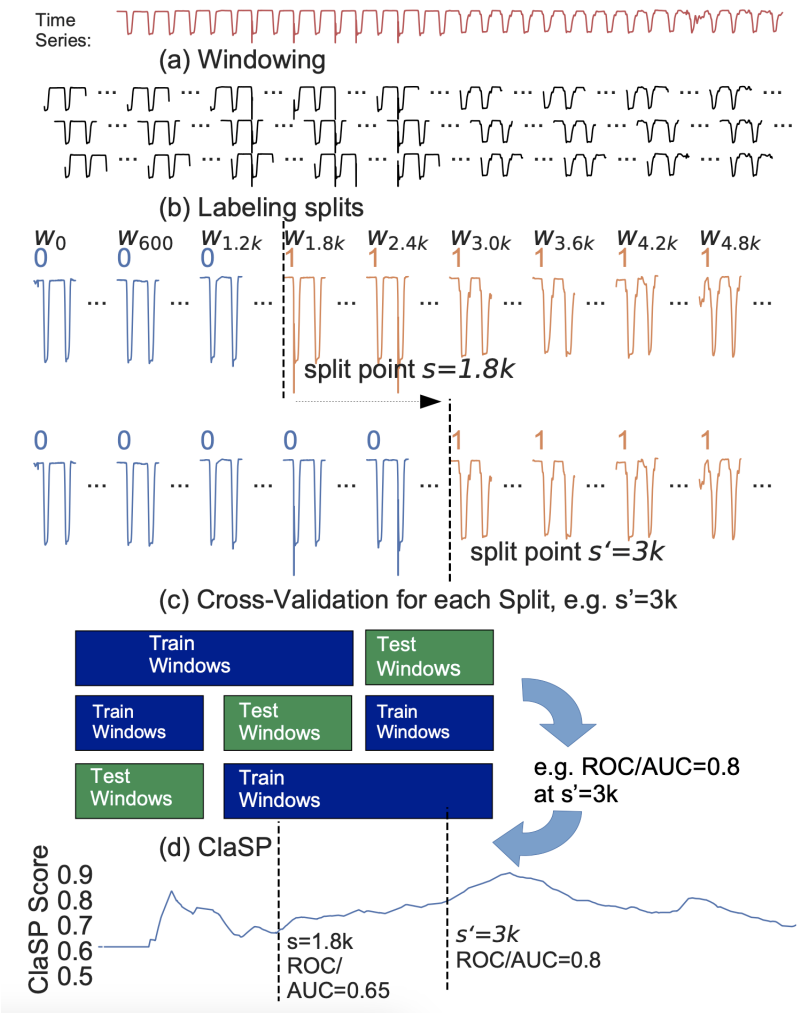

Abbildung 1: Beispielhaftes Vorgehen von ClaSP, bei dem der Algorithmus zuerst eine Zeitreihe in Fenster unterteilt. Anschließend wird das Finden der kNN und Relabeling hypothetischer Änderungspunkte durchgeführt, welches durch Leave-One-Out Cross Validation zum Classification Score Profile führt. [11]

3.1 Classification Score Profile (ClaSP)

ClaSP versucht das Problem der Zeitreihensegmentierung zu lösen. Dazu wird das namensgebende Classification Score Profile berechnet. Um dies zu erhalten, wird eine Zeitreihe T der Länge n zunächst in überlappende, gleich große Teilsequenzen mit fixer Länge w partitioniert. Daraus resultieren $n - w + 1$ Teilsequenzen, die auch als Fenster bzw. Windows bezeichnet werden. Abbildung 1 (a) zeigt die aus einer gegebenen Zeitreihe berechneten überlappenden Fenster.

Im Anschluss werden zu jedem dieser Fenster einmalig die k nächsten Nachbarn

berechnet. Die kNN erhalten wir durch eine Distanzmatrix, die mit dem STOMP-Algorithmus [18] berechnet wird. Aus dieser Matrix werden die kNN entnommen. Daraufhin werden mehrere selbst-überwachte binäre Klassifikationsprobleme mithilfe der kNN berechnet. Dabei iteriert ClaSP durch die Zeitreihe und setzt hypothetische Änderungspunkte. Zu jedem hypothetischen Änderungspunkt werden alle Fenster, die links des Änderungspunktes liegen mit null gelabelt, alle rechts davon mit eins. Abbildung 1 (b) zeigt zwei hypothetische Änderungspunkte mit dem Fenster 1800 und 3000 mit dem dazugehörigen Labeling, wobei alle Fenster links der Änderungspunkte mit null gelabelt werden (blau) und alle rechts davon mit eins (orange). Dieser Vorgang wird auf jedes Fenster angewendet, sodass die erste Position jedes Fensters einen hypothetischen Änderungspunkt darstellt.

Während dieses Vorgangs wird jedes Labeling mithilfe einer Leave-One-Out Cross Validation und einer festgelegten Bewertungsfunktion evaluiert, indem die Label der kNN eines Fensters mit dem Label des Fensters verglichen werden. Das zu vergleichende Label der kNN innerhalb eines Fensters wird per Mehrheitsentscheid festgelegt. Abbildung 1 (c) zeigt exemplarisch die Leave-One-Out Cross Validation mit dem Änderungspunkt an Position 3000.

Das Klassifikationsergebnis jeder Cross Validation zu den hypothetischen Änderungspunkten bilden das Classification Score Profile (ClaSP). Daraufhin bestimmt der Index des Maximums von ClaSP den Änderungspunkt zweier Segmente. Abbildung 1 (d) zeigt, wie die einzelnen Ergebnisse der Cross Validation als Folge aufgezeigt werden. Folglich verfeinert der Algorithmus die Segmentierung der Zeitreihe rekursiv mit weiteren Änderungspunkten, indem er die vorherigen Schritte erneut durchführt. [11]

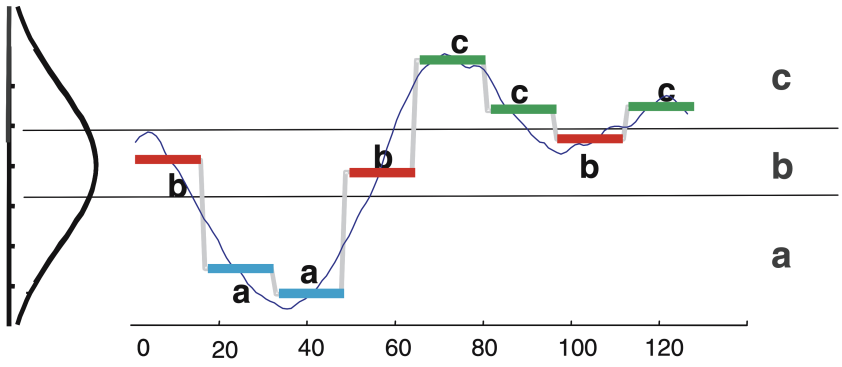

Abbildung 2: Beispielhaftes Vorgehen von SAX, bei dem generierte PAA-Koeffizienten mithilfe der Breakpoints auf ein Wort baabccbc abgebildet werden [7].

3.2 Symbolic Aggregate approXimation (SAX)

SAX ist eine symbolische Repräsentation einer Zeitreihe [7]. Es approximiert und diskretisiert dazu eine Zeitreihe der Länge n auf einen String mit kürzerer Länge w.

Der erzeugte String wird aus einem Alphabet mit festgelegter Größe gebildet. Um die approximierte Zeitreihe zu erhalten, normalisieren und transformieren wir die Zeitreihe zunächst in die sogenannte Piecewies Aggregate Approximation (PAA). PAA reduziert die Zeitreihe der Länge n in w gleich große Frames. Aus den Datenpunkten, die sich innerhalb eines Frames befinden, wird der Durchschnitt gebildet [2]. Dies wird PAA-Koeffizient genannt. Mithilfe von PAA können wir diskrete Werte berechnen und die gegebene Zeitreihe durch einen String repräsentieren [7]. Abbildung 2 zeigt hierbei die verschiedenen Schritte der Transformation auf.

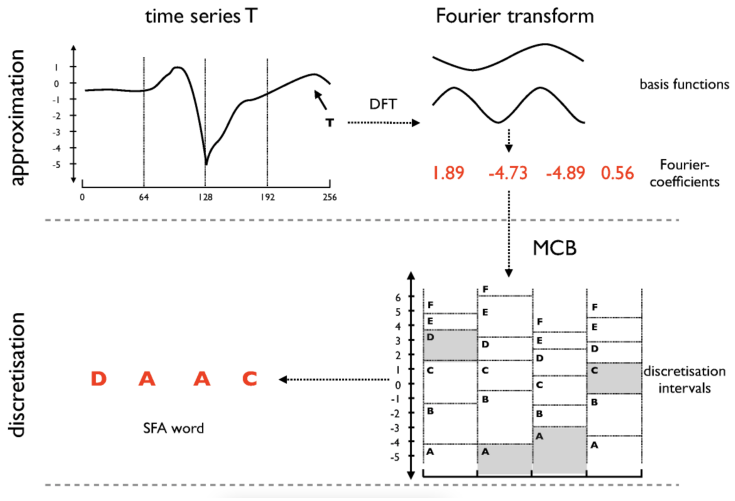

Abbildung 3: Beispielhaftes Vorgehen von SFA, bei dem eine Zeitreihe durch DFT approximiert und im Anschluss durch MCB zu dem SFA-Wort DAAC diskretisiert wird [12].

3.3 Symbolic Fourier Approximation (SFA)

SFA ist ebenfalls eine symbolische Repräsentation einer Zeitreihe[12]. Dabei wird die Zeitreihe in gleitende Fenster unterteilt, wobei die Datenpunkte eines jeden Fensters in sogenannte SFA-Wörter diskretisiert werden. Diese Wörter bestehen aus einem endlichen Alphabet und verfügen über eine festgelegte Länge. Zum Erlangen der SFA-Wörter werden drei Schritte benötigt: Zunächst die Zerlegung der Zeitreihe in überlappende Fenster. Anschließend werden die Fenster mittels Diskreter Fourier-Transformation (DFT) approximiert und durch Multi Coefficient Binning (MCB) diskretisiert [12]. Abbildung 3 zeigt hierbei ein beispielhaftes Vorgehen der Approximation und Diskretisierung.

SAX und SFA werden dazu beitragen, die Implementierung mit dem STOMP-Algorithmus zur Ermittlung der k nächsten Nachbarn in ClaSP zu ersetzen.

4 Methodik

Die in [4] bereitgestellte Python-Implementation von ClaSP soll mit dieser Bachelorarbeit angepasst und evaluiert werden. Dabei wird der STOMP-Algorithmus [18] zur Ermittlung der kNN durch die Verwendung von SAX mit [15] und SFA mit [16] ersetzt, die jeweils durch das sktime-Framework bereitgestellt werden. Sktime ist eine Open-Source Python-Softwarebibliothek, die maschinelle Lernmodelle für Zeitreihen zur Verfügung stellt [14]. Es werden zwei Varianten für die Ermittlung der kNN implementiert. Eine Variante, die SAX verwendet und eine, die SFA verwendet. Dies ermöglicht es im Evaluationsteil der Bachelorarbeit, die unterschiedlichen Resultate bezüglich Laufzeit und Genauigkeit zu diskutieren.

Da die Fenster einer Zeitreihe mithilfe von SAX oder SFA durch symbolische Wörter approximiert werden, bietet es sich an, die Positionen der gewonnenen Wörter mithilfe eines Inverted Index zu speichern. Dabei werden die Key-Value Paare aus den Wörtern mit deren vorkommenden Positionen bestehen. Mithilfe dieses Inverted Index können daraufhin auf effiziente Weise für jedes approximierte Wort aus der Zeitreihe die einzelnen Positionen gefunden werden und damit zu einer Verschnellerung von ClaSP führen. Dazu wird als Distanzfunktion für die kNN-Approximation das direkte Matching von approximierten Wörtern benutzt. Das heißt, es werden nur Positionen der Wörter für das Matching in Betracht gezogen, die mit dem Wort übereinstimmen, für das die k nächsten Nachbarn approximiert werden sollen. Das direkte Matching wird unter der Annahme verwendet, dass sich Teilsequenzen einer Zeitreihe häufig wiederholen und es ausreicht, ähnliche Teilsequenzen im gleichen Segment als kNN zu finden. Dadurch gibt es für ein gegebenes Wort eine variable Anzahl approximierter nächster Nachbarn, aus denen k ausgewählt werden müssen. Dazu kann man einen kNN wie folgt approximieren. Eine Möglichkeit, einen kNN zu approximieren, ist die ersten k Positionen zu einem Wort aus dem Inverted Index als die nächsten Nachbarn zu bestimmen. Alternativ dazu könnten die k Wörter verwendet werden, welche in der Zeitreihe am nächsten zu einem Fenster liegen. Eine dritte Möglichkeit wäre, k zufällige Fenster zu bestimmen. Die dadurch zu jedem Wort gefundenen kNN Positionen können dabei in einer Matrix gespeichert werden, welche für die weiteren Schritte von ClaSP verwendet wird.

Die beiden Implementierungen durch SAX und SFA werden hier mit der Absicht verwendet, die Suche nach den approximierten k nächsten Nachbarn der einzelnen Positionen in (log)-linearer Laufzeit in Abhängigkeit von der Zeitreihenlänge durchzuführen, verglichen mit der aktuellen Version mit einer Laufzeit von $O(n^2)$ für die tatsächlichen kNN [11]. Durch die Approximation und den darin eingehenden Verlust an Informationen ist es dabei jedoch auch möglich, eine geringe Genauigkeit der zu findenden Wechselpunkte in der Zeitreihe zu erhalten. Es wird jedoch angestrebt, den Verlust der Genauigkeit möglichst gering zu halten.

5 Evaluation

Nachdem die beiden neuen Implementierungen zum Finden der k nächsten Nachbarn durch SAX und SFA implementiert wurden, sollen deren Laufzeit und Genauigkeit miteinander und mit der bisherigen Implementierung verglichen werden. Dazu werden wie auch in [11] die vorhandenen 98 Benchmark-Datensätze zum Vergleich der drei Implementierungen herangezogen. Diesbezüglich sollen die Segmentierungen einiger Zeitreihen explorativ analysiert werden, um einen direkten Vergleich zu bisherigen Ergebnissen zu erhalten. 32 der 98 Datensätze stammen aus öffentlichen Segmentierungs-Benchmark-Datensätzen [6], die biologische, mechanische oder synthetische Prozesse erfassen. Die übrigen 66 Datensätze wurden aus einem Teil der insgesamt 120 Datensätze des UCR Archivs [3] entnommen. Dabei verfügt jeder Datensatz über offensichtliche Periodizitäten. Zu jedem der 98 Datensätze liegen Änderungspunkte vor, welche von Experten festgelegt wurden. Insgesamt haben 49 Datensätze 2 Segmente, 22 Datensätze haben 3 Segmente, 10 Datensätze haben 4 Segmente, 11 Datensätze haben 5 Segmente, 1 Datensatz hat 6 Segmente und 5 Datensätze haben 7 Segmente. Zudem verfügen alle Datensätze über eine festgelegte Fenstergröße, die von ClaSP genutzt wird [11].

Diese Arbeit sieht vor, Default Parameter für die neuen Implementierungen mit SAX und SFA zu finden. Dies können beispielsweise die Größe von k für die kNN, die Fenstergröße, die Wortlänge oder die Alphabetgröße sein. Die Evaluationsmetrik, definiert in [11], bewertet dabei die angewendeten Verfahren. Diese Metrik summiert und normalisiert die relativen Abstände zwischen jedem vorhergesagten Änderungspunkt p und dem dazu nächstgelegensten Ground Truth Änderungspunkt [11]. Dabei wird diese Metrik auch in ClaSP verwendet [11] und hilft uns Vergleiche zu den verschiedenen Implementierungen mit SAX und SFA von ClaSP zu ermöglichen. Darüber hinaus werden die Implementierungen mit den Kompetitoren Autoplait [9], FLOSS [6], Binary Segmentation (BinSeg) [13] und Bayesian Online Changepoint Detection (BOCD) [1] aus dem Clasp Paper [11] verglichen. Die erzielten Ergebnisse werden abschließend durch CD-Diagrammen, Boxplots und Tabellen präsentiert.

Literatur

[1] Ryan Prescott Adams und David JC MacKay. "Bayesian online changepoint detection". In: *arXiv preprint arXiv:0710.3742* (2007).

[2] Kaushik Chakrabarti u. a. "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases". In: *ACM Trans. Database Syst.* 27.2 (Juni 2002), S. 188–228. ISSN: 0362-5915. DOI: 10.1145/568518.568520. URL: https://doi.org/10.1145/568518.568520.

[3] Yanping Chen u. a. *The UCR Time Series Classification Archive.* www.cs.ucr.edu/~eamonn/time_series_data/. Juli 2015.

[4] *ClaSP Code and Raw Results.* 2021. URL: https://sites.google.com/view/ts-clasp/ (besucht am 20. 10. 2022).

[5] OE Dick u. a. “Analysis of EEG patterns in subjects with panic attacks”. In: *Human Physiology* 46.2 (2020), S. 163–174.

[6] Shaghayegh Gharghabi u. a. “Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels”. In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, S. 117–126. DOI: 10.1109/ICDM.2017.21.

[7] Jessica Lin u. a. “Experiencing SAX: A Novel Symbolic Representation of Time Series”. In: *Data Min. Knowl. Discov.* 15 (Aug. 2007), S. 107–144. DOI: 10.1007/s10618-007-0064-z.

[8] Miodrag Lovrić, Marina Milanović und Milan Stamenković. “Algoritmic methods for segmentation of time series: An overview”. In: *Journal of Contemporary Economic and Business Issues* 1.1 (2014), S. 31–53.

[9] Yasuko Matsubara, Yasushi Sakurai und Christos Faloutsos. “AutoPlait: Automatic Mining of Co-Evolving Time Sequences”. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’14. Snowbird, Utah, USA: Association for Computing Machinery, 2014, S. 193–204. ISBN: 9781450323765. DOI: 10.1145/2588555.2588556. URL: https://doi.org/10.1145/2588555.2588556.

[10] Michael JAM van Putten. “Nearest neighbor phase synchronization as a measure to detect seizure activity from scalp EEG recordings”. In: *Journal of clinical neurophysiology* 20.5 (2003), S. 320–325.

[11] Patrick Schäfer, Arik Ermshaus und Ulf Leser. “ClaSP - Time Series Segmentation”. In: *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*. CIKM ’21. Virtual Event, Queensland, Australia: Association for Computing Machinery, 2021, S. 1578–1587. ISBN: 9781450384469. DOI: 10.1145/3459637.3482240. URL: https://doi.org/10.1145/3459637.3482240.

[12] Patrick Schäfer und Mikael Högqvist. “SFA: A Symbolic Fourier Approximation and Index for Similarity Search in High Dimensional Datasets”. In: *Proceedings of the 15th International Conference on Extending Database Technology*. EDBT ’12. Berlin, Germany: Association for Computing Machinery, 2012, S. 516–527. ISBN: 9781450307901. DOI: 10.1145/2247596.2247656. URL: https://doi.org/10.1145/2247596.2247656.

[13] A. J. Scott und M. Knott. “A Cluster Analysis Method for Grouping Means in the Analysis of Variance”. In: *Biometrics* 30.3 (1974), S. 507–512. ISSN: 0006341X, 15410420. URL: http://www.jstor.org/stable/2529204 (besucht am 27. 10. 2022).

[14] *Sktime Python Framework*. URL: https://www.sktime.org/en/stable/ (besucht am 20. 10. 2022).

[15] *Sktime Python SAX Implementierung.* URL: https://www.sktime.org/en/stable/api_reference/auto_generated/sktime.transformations.panel.dictionary_based.SAX.html (besucht am 20. 10. 2022).

[16] *Sktime Python SFA Implementierung.* URL: https://www.sktime.org/en/stable/api_reference/auto_generated/sktime.transformations.panel.dictionary_based.SFA.html (besucht am 20. 10. 2022).

[17] Jiangling Yin, Yain-Whar Si und Zhiguo Gong. "Financial time series segmentation based on turning points". In: *Proceedings 2011 International Conference on System Science and Engineering*. IEEE. 2011, S. 394–399.

[18] Y. Zhu u. a. "Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins". In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. Los Alamitos, CA, USA: IEEE Computer Society, Dez. 2016, S. 739–748. DOI: 10.1109/ICDM.2016.0085. URL: https://doi.ieeecomputersociety.org/10.1109/ICDM.2016.0085.