FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Seminararbeit:

Erkennung von Anomalien in Zeitreihen durch One Class SVM und K-Means Clustering

19.12.2024

Aachen

Verfasser: Rashid Goulebe

Erstprüfer: Prof. Stephan Bialonski

Zweitprüfer: Joel Otte

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

Erkennung von Anomalien in Zeitreihen durch

One Class SVM und K-Means Clustering

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Name: Rashid Goulebe

Aachen, den 19.12.2024

Unterschrift der Studentin / des Studenten

Zusammenfassung

Diese Seminararbeit beschäftigt sich damit, wie mit Hilfe der beiden Lernalgorithmen „K-Means Clustering“ und „One Class SVM“ Anomalien in Zeitreihen erkannt werden können. Durchgeführt werden diese Verfahren an einem simulierten Datensatz, welcher die Besucher einer Webseite pro Tag abbilden soll. In den originalen Daten wurden die Anomalien bereits markiert. Diese Information wird lediglich bei der Auswertung und Interpretation der Ergebnisse mit dem F1-Score verwendet und kommt nicht beim Training der Lernalgorithmen zum Einsatz.

Zur Interpretation der Datenpunkte mit den Algorithmen wird die Zeitreihe zunächst in Test- und Trainingsdaten unterteilt und mit dem „Windowing“-Verfahren in kleinere Teilsegmente zerlegt. Anschließend werden diese Teilsegmente durch die beiden Algorithmen interpretiert. Nach der Interpretation werden die Segmente wieder zurück in ihre Ursprungsform rücktransformiert und mit den original gelabelten Daten abgeglichen. Als Maß der Genauigkeit wird hier der F1-Score verwendet

Abschließend wird ein Fazit zu den beiden Lernalgorithmen, in Kombination mit dem Windowing-Verfahren gezogen und Optimierungsvorschläge für zukünftige Verbesserungen gegeben.

Inhalt

Einleitung

Die Analyse von Zeitreihen befasst sich mit der Untersuchung von Datenpunkten, die in zeitlicher Reihenfolge gesammelt wurden. Ziel hierbei ist es, Muster zu erkennen, Vorhersagen zu treffen und zu verstehen, wie verschiedene Komponenten interagieren.

Zeitreihen und deren Analyse beschäftigen die Menschen schon seit tausenden von Jahren. Eine der ersten Formen der Dokumentation von aufeinanderfolgenden Messwerten lässt sich bis nach Mesopotamien um das Jahre 3000 v. Chr. zurückführen. Hier wurden erste Bilder und Symbole benutzt, um landwirtschaftliche Listen und Tabellen anzufertigen, welche die Ernteerträge darstellten. So konnten die Schwankungen in den Erträgen über mehrere Erntejahre hinweg besser nachvollzogen und ins Verhältnis gesetzt werden. [1]

Das Erfassen von Messwerten mit zeitlicher Komponente findet zur heutigen Zeit beinahe in jedem Bereich einen wichtigen Anwendungsfall. Beispielsweise werden in der Medizin die elektrischen Aktivitäten von Herzmuskelfasern über einen bestimmten Zeitraum erfasst und zu einem Elektrokardiogramm (kurz EKG) zusammengefasst. Ein weiteres Beispiel ist die Meteorologie, bei der die Analyse von Temperaturen und Luftfeuchtigkeit sogar zu einem gewissen Grad die zukünftigen Wetterverhältnisse vorhersagen kann.

Auch die e-dynamics GmbH analysiert als Dienstleisterin für ihre Kunden verschiedene Zeitreihen. Hierbei handelt es sich hauptsächlich um Informationen über das Nutzungsverhalten von Usern auf Webseiten. Durch diese Analyse kann beispielsweise die „Conversion Rate“ ermittelt werden, bei der die Anzahl der Besucher ins Verhältnis zur Anzahl der Kaufabschlüsse gebracht wird. Diese Daten sind zum größten Teil zeitbasiert und lassen sich daher in Zeitreihen darstellen.

Zusätzlich entwickelt e-dynamics ein neues Tool (ed.Detect), welches basierend auf den Daten über das Nutzungsverhalten einer Website bestimmte Anomalien erkennt. Bei diesen Anomalien kann es sich beispielsweise um Bot Traffic[1] handeln, wenn ungewöhnlich viele Nutzer die Seite besuchen. Ein plötzlicher Einbruch der Besucher einer Website könnte darauf hindeuten, dass die Seite nicht zu erreichen war oder eine Fehlfunktion hatte.

Die Definition von „Anomalie Erkennung“ ist die Erkennung von Datenpunkten in einem Datensatz oder einem System, die von der Norm abweichen. Hierbei ist zu beachten, dass es verschiedene Kategorien von Anomalien gibt. [2]

> **Punktanomalien** bezeichnen einzelne Datenpunkte, welche vom Rest des Datensatzes signifikant abweichen. Ein Beispiel für eine Punktanomalie ist ein plötzlich hoher Einkaufswert eines Kunden, der für den jeweiligen Einzelhandel von der Norm abweicht. Auch wenn diese Anomalie für den Einzelhändler erfreulich ist, muss es als Anomalie markiert werden. Ein Grund hierfür könnte sein, dass sich der Kunde bei der Mengeneingabe vertippt hat.
>
> **Kontextbezogene Anomalien** stehen im Zusammenhang mit weiteren Informationen, die ein gewisses Verhalten erwarten. Ein Unternehmen, das am Black Friday den umsatzstärksten Tag des Jahres verzeichnet muss diesen nicht zwangsläufig als Anomalie markieren.

[1] automatisierter Datenverkehr auf einer Website oder einem Netzwerk, der von Software-Robotern (Bots) generiert wird und sowohl nützliche Zwecke (z. B. Suchmaschinen-Crawling) als auch schädliche Aktivitäten (z. B. Spam oder DDoS-Angriffe) umfassen kann.

Zeitliche Anomalien und **Zeitreihenanomalien** beziehen sich auf die Abweichung eines zeitlich wiederkehrenden oder auch saisonalen Musters. Ein Beispiel hierfür wäre der Tag-Nacht-Rhythmus. Eine Website auf der z.B. Kleidung in Deutschland verkauft wird, lässt vermuten, dass die Anzahl der Aufrufe tagsüber höher ist als nachts. Eine Abweichung von diesem Muster kann als Anomalie interpretiert werden.

Zur Erkennung dieser anormalen Datenpunkte können beispielsweise maschinelle Lernalgorithmen eingesetzt werden, welche die betroffenen Datenpunkte markiert. So können potenzielle Risiken, Effizienzmängel oder gezielte Angriffe auf Webseiten schnell erkannt werden und ermöglichen damit eine höhere Reaktionszeit für Gegenmaßnahmen.

Diese Seminararbeit behandelt die Erkennung von Anomalien mit **K-Means-Clustering** und **One Class SVM.** Beide Lernalgorithmen zählen zu der Kategorie „Unüberwachtes Lernen“. Unter diesen Begriff fallen alle Arten von maschinellem Lernen, bei denen das Ergebnis unbekannt ist und es keine gelabelten Daten zum Trainieren des Algorithmus gibt. Der Lernalgorithmus erhält demnach lediglich die Eingabedaten und wird damit beauftragt, Wissen aus diesen Daten zu extrahieren. [3] Diese Seminararbeit geht zunächst auf den Datensatz ein, sowie auf dessen Umformung der Daten in eine Form, die für die Interpretation der Daten durch die Lernalgorithmen sinnvoller ist. Anschließend betrachten wir im Einzelnen die beiden maschinellen Lernalgorithmen K-Means Clustering und One Class SVM zur Erkennung von Anomalien.

Der Datensatz

Wie oben erwähnt arbeitet e-dynamics mit Nutzerdaten von Websitebesuchern. Diese Daten können beispielsweise die Anzahl der Besucher einer Website, das Benutzen eines Buttons oder die Tätigung eines Kaufes sein. Typischerweise werden diese Nutzerdaten unter dem Sammelbegriff „Web-Tracking“ zusammengefasst und haben meistens einen Zeitstempel.

Das Besondere an den Daten ist zum einen ihre saisonalen Schwankungen, welche von ganz unterschiedlicher Periodenlänge sein können. Im Allgemeinen beziehen sich saisonale Schwankungen auf ein vorhersehbares und wiederkehrendes Muster, das über einen bestimmten Zeitraum auftreten kann. [4] Die Schwankungen können sich unter anderem an Tages-, Wochen-, Monats- oder Jahres-Rhythmen orientieren. Die Gründe für solche Schwankungen im Datensatz können beispielsweise der Schlafrhythmus von Konsumenten für eine tägliche Schwankung oder eine steigende Nachfrage an Weihnachten für eine jährliche Schwankung sein.

Eine weitere Komponente der Web-Analytics Daten zeichnet sich durch sogenanntes Rauschen aus. Unter statistischem Rauschen versteht man zufällige Schwankungen in statistischen Daten. Diese Schwankungen haben nichts mit den tatsächlichen Beziehungen in den Daten zu tun. Dieses Rauschen kann durch eine Vielzahl von Faktoren wie Messfehler, Datenmanipulation oder zufällige Schwankungen verursacht werden. [5]

Speziell im Bereich Web-Analytics kann dieses Rauschen beispielsweise durch eine fehlerhafte Implementierung des Web-Tracking auf der Website oder durch Internetprobleme der Benutzer auftreten. Ebenfalls blockieren gewisse Browser oder Browser-Extensions sämtliche Analytics-Systeme und damit das Web-Tracking, was das reale Mesergebnis verfälscht.

Da die Daten unserer Kunden nicht veröffentlicht werden dürfen, hat das Data-Analytics Team speziell für diese Seminararbeit einen Datensatz synthetisch generiert. Der Datensatz repräsentiert die Anzahl der Besucher einer imaginären Homepage einer Website.

Der Zeitraum der simulierten Messung erstreckt sich über 4 Jahre und summiert täglich alle Besucher des vergangenen Tages auf. Die Datensätze sind mit geringem Rauschen kontaminiert, welches die Messfehler imitiert. Die Anomalien, die der Datensatz beinhaltet sollen unerwünschten Bot-Traffic, sowie Seitenausfälle simulieren. Zur besseren Validierung der Daten wurden die Anomalien im Datensatz markiert (siehe Abbildung 1). Beim späteren Training werden diese Informationen allerdings nicht benutzt, da es sich bei den Lernalgorithmen um unüberwachtes Lernen handelt.

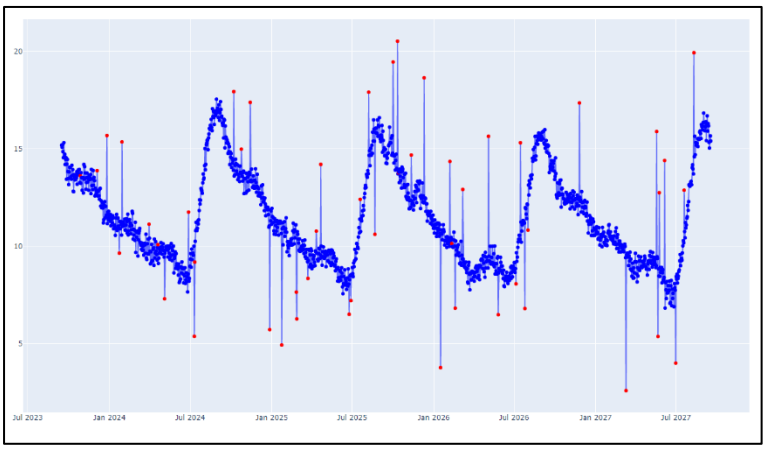

Abbildung 1: VIsualisierung des Datensatzes mit roter Markierung der Anomalien

Der Datensatz wird nach der Hold-Out Validierung aufbereitet. Hierbei wird der Datensatz mit N Datenpunkten in Trainingsdaten D_{train} (N-K Datenpunkte) und in Validierungsdaten D_{val} (K Datenpunkte) mit K = N*0,2 unterteilt. Basierend darauf wird eine Gittersuche durchgeführt, welche die optimalen Hyperparameter ermittelt und so die perfekte Einstellung für die Algorithmen findet.

Windowing

Der Datensatz liegt aktuell in einer Form vor, die zur Interpretation mit den ausgewählten Algorithmen nicht praktikabel ist. Daher wird in dieser Seminararbeit das Sequenzierungsverfahren namens „Windowing“ angewandt, welches den Datensatz in 3 Schritten von einer Zeitreihe in ein n-Dimensionales Cluster transformiert. Der folgende Abschnitt beleuchtet die einzelnen Schritte im Detail.

Segmentierung

Bei der Segmentierung wird ein Fenster (Window) der Größe n über den Datensatz gelegt und um den Wert m verschoben. So entstehen überlappende Segmente, die jeweils einen Teil der Kurve abbilden und darstellen. Die entstandenen Segmente haben dieselbe Länge wie das Fenster (n). Der Grad der Überlappung wird mit dem Parameter m festgelegt, welcher für die Verschiebung des Fensters verantwortlich ist.

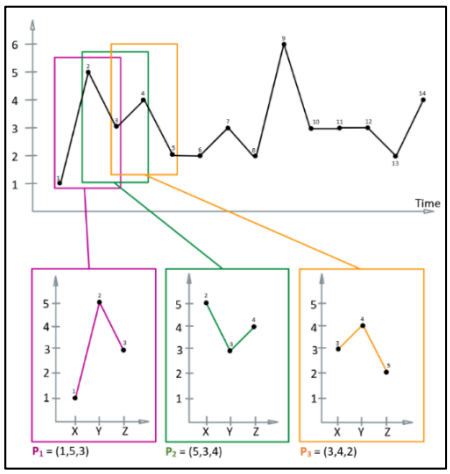

Abbildung 2: Windowing einer Zeitreihe mit Fenstergröße 3 und einer Verschiebung von 1

In unserem Beispiel wählen wir aus Veranschaulichungsgründen eine Fenstergröße n = 3 und eine Verschiebung m = 1. Damit rutscht das Fenster immer um einen Datenpunkt weiter und wir erhalten eine hohe Überlappung der Segmente. Diese Segmente werden im nächsten Schritt in ein N-Dimensionales Koordinatensystem überführt und mit

den oben genannten Algorithmen interpretiert. Dabei geht es vor allem um Ähnlichkeiten der einzelnen Kurvenformen.

Abbildung 2 zeigt im oberen Graphen eine beispielhafte Zeitreihe mit 14 Datenpunkten. Die farbig markierten Kästen zeigen die einzelnen Fenster der Größe n = 3, die jeweils um einen Datenpunkt verschoben werden. Der untere Teil der Abbildung stellt die daraus resultierenden Segmente mit jeweils drei Datenpunkten dar. Die perfekte Fenstergröße sowie die ideale Verschiebung sind zwei von vielen Parameter, die im Laufe der Seminararbeit verändert und vergleichen werden.

Die daraus gewonnenen Segmente können nun, wie ein n-Dimensionales Cluster interpretiert werden. Da wir in diesem Beispiel eine Fenstergröße von n=3 gewählt haben, resultiert daraus auch eine Segmentlänge von drei. Daraus folgt, dass die Daten in ein dreidimensionales Koordinatensystem übertragen werden können.

Abbildung 3 zeigt im oberen Teil die einzelnen Segmente, die durch das Windowing-Verfahren ermittelt wurden. Unter den einzelnen Segmenten stehen die Punkte die sie in einem $\mathbb{R}^3$dimensionalen Koordinatensystem repräsentieren(P_1=(1,5,3), ...).

Der untere Teil von *Abbildung 3* stellt den Graphen dar, in dem die Segmente überführt werden können. Ziel der Umwandlung in diese Form ist es, ähnliche Kurvenformen in ähnlichen Bereichen des Grafen wiederzufinden. So können Cluster gebildet werden, die ähnliche Kurvenformen unter sich vereinen.

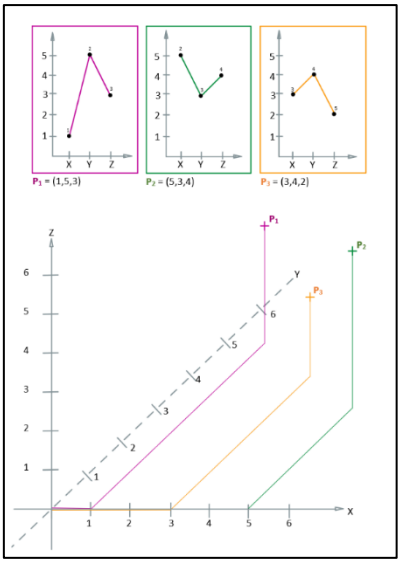

Abbildung 3: Einordnung der Segmente aus Abbildung 2 in ein dreidimensionales Koordinatensystem

Bei einem realen Datensatz mit einer größeren Fenstergröße werden die Segmente dementsprechend in ein n-dimensionales Koordinatensystem übertragen. Da dies allerdings nicht anschaulich darzustellen ist, sind die Segmente in diesem Beispiel von der Länge n = 3. [6]

Diese Cluster werden mit Hilfe der Maschine Learning Algorithmen gefunden und interpretiert. Im nächsten Abschnitt wird genauer auf die Anwendung der Algorithmen auf die einzelnen Segmente eingegangen.

Anwendung auf den Datensatz

Wenn wir diese Schritte auf unseren Datensatz anwenden, so erhalten wir ebenfalls die dreidimensionale Grafik, die wir im vorherigen Kapitel hergeleitet haben. Abbildung 4 zeigt die Visualisierung der segmentierten Trainingsdaten. Hierbei ist zu beachten, dass ein Punkt rot (als Anomalie) markiert wird, sobald einzelne Punkte in dem Segment als Anomalie gelten. Die Tatsache, dass so nur ganze Segmente statt einzelner Datenpunkte als Anomalien markiert werden, wird vor allem in der Validierung der Lernalgorithmen eine Rolle spielen.

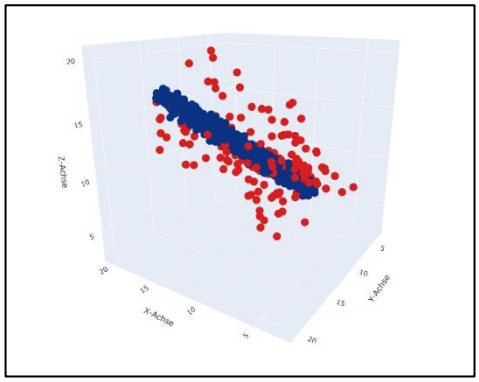

Abbildung 4: Segmentierung der Trainingsdaten und Visualisierung in dreidimensionalem Raum

Rücktransformation

Die Tatsache, dass Segmente als Anomalie interpretiert werden, obwohl eventuell nur ein Datenpunkt des Segmentes eine Anomalie ist, stellt uns vor eine Herausforderung. Abbildung 5 verdeutlicht dieses Problem. Hier werden die einzelnen Schritte der Transformation visualisiert und die Fehlerfortpflanzung dargestellt.

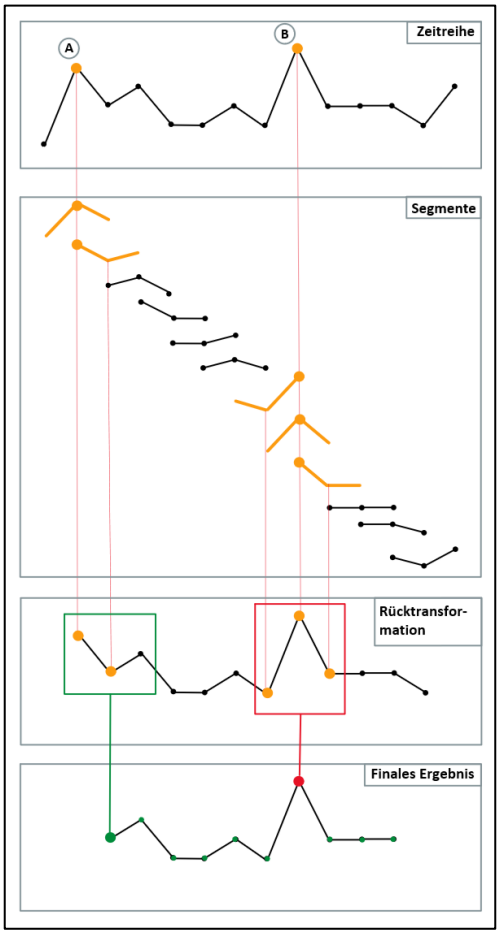

Abbildung 5: Transformierung und Rücktransformierung der Zeitreihe

Der oberste Graph der Abbildung mit der Beschriftung „Zeitreihe“ stellt die Ausgangssituation dar. Die orangenen Punkte mit der Beschriftung „A“ und „B“ stellen größere Ausreißer im Datensatz dar. Auf diese Zeitreihe wird im nächsten Schritt die Segmentierung durch das Windowing-Verfahren mit der Fenstergröße 3 und einer Verschiebung von 1 angewandt.

Die Grafik mit der Überschrift „Segmente“ zeigt die daraus resultierenden Segmente, die in dieser Form von den verschiedenen Lernalgorithmen interpretiert werden können. Durch die Fenstergröße 3 könnten die Segmente hier auch wieder in ein dreidimensionales Koordinatensystem übertragen werden. Wichtig ist hierbei zu beachten, dass Segmente mit größeren Abweichungen bzw. potenziellen Anomalien sich dezentral lokalisieren. Genau diese Eigenschaft machen wir uns bei der Erkennung von Anomalien zu nutze. Das Problem hierbei ist allerdings, dass nur ganze Segmente damit als Anomalie markiert werden können. Die orange markierten Segmente in der Grafik „Segmente“ visualisiert beispielhaft, welche Segmente von den Lernalgorithmen als Ausreißer markiert werden würden. Jedes Segment, das an einer Stelle eine Anomalie enthält, wird als Ausreißer interpretiert.

Wenn diese Informationen wieder auf die Zeitreihe anwendet werden, ergeben sich mehr Ausreißer als ursprünglich vorhanden. (siehe Abbildung 5, Grafik Rücktransformation).

Um diesen Fehler zu beheben, gibt es verschiedene Lösungsansätze. Ein Ansatz wäre von Anfang an immer genau so viele Datenpunkte weiter zu rutschen, wie das Fenster groß ist. So kommt jede Anomalie nur ein einziges Mal in einer Kurve vor.

In dieser Seminararbeit wird allerdings ein anderer Ansatz betrachtet. Nach der Rücktransformation der Daten wird erneut ein Sliding Window auf den Datensatz angewandt. Dieses Fenster hat dieselbe Größe wie das Fenster der Segmentierung. Innerhalb des neuen Fensters überprüfen wir, ob alle Werte als Anomalie gekennzeichnet wurden (siehe Abbildung 5, Rücktransformation) Nur wenn dies der Fall ist, wird der Fehler in den „bereinigten“ finalen Graphen übertragen.

F1-Score zur Validierung der Daten

Um den Algorithmus validieren und beurteilen zu können, nutzt diese Seminararbeit den F1-Score. Dieser bietet sich an, da die Bewertung der Daten binärer Natur ist (Anomalie oder keine Anomalie). Die Kennzahl (F1-Score) stellt das gewichtete harmonische Mittel aus Präzision und Recall dar. Diese beiden Zustände sind abhängig davon, ob die Klassifikation der Lernalgorithmen wirklich korrekt ist:

True Positive: Der Algorithmus klassifiziert den Datenpunkt als Anomalie und es ist tatsächlich eine Anomalie.
False Positive: Der Algorithmus behauptet der Datenpunkt wäre eine Anomalie, dabei handelt es sich aber nicht wirklich um eine Anomalie.
True Negative: Der Algorithmus erkennt richtig, dass es sich nicht um eine Anomalie handelt.
False Negative: Der Algorithmus behauptet fälschlicherweise, dass es sich nicht um eine Anomalie handeln würde.

$$F1 = \frac{2 * Precision * Recall}{Precission + Recall}$$

Der Scope des F1-Scores kann Werte zwischen 0 und 1 annehmen, wobei ein hoher Wert für ein aussagekräftiges Modell mit einer hohen Präzision und einem hohen Recall steht.

Die Präzision misst in einer Klassifizierung den Anteil der positiven Vorhersagen, die korrekt klassifiziert wurden, im Vergleich zur Gesamtzahl der positiven Fälle im Datensatz. Ein hoher Wert der Präzision bedeutet, dass ein Großteil der als Anomalie deklarierten Datenpunkte auch wirklich Anomalien waren und nur wenige Datenpunkte fälschlicherweise als Anomalie klassifiziert werden.

$$Precision = \frac{\text{True Positive}}{True\ Positive\ +\ False\ Positive}$$

Der Recall (im Deutschen auch Sensitivität) beschreibt die Fähigkeit die wichtigen Instanzen in einem Datensatz richtig vorherzusagen. Dabei wird das Verhältnis zwischen den korrekt vorhergesagten positiven Instanzen (True Positives) und der Gesamtzahl aller tatsächlich positiven Instanzen (True Positive + False Negative) berechnet. Dadurch wird mit dieser Kennzahl auch gemessen, ob sich die Anzahl der False Negatives im Rahmen hält, was bei der Präzision nicht beachtet wird.

$$Recall = \frac{\text{True Positives}}{True\ Positives\ +\ False\ Negatives}$$

Der Recall ist vor allem in Anwendungen wichtig, in denen die Identifizierung aller positiven Instanzen im Fokus steht und dabei auch falsche Vorhersagen in Kauf genommen werden können. In der Medizin beispielsweise ist es wichtiger, alle erkrankten Menschen zu erkennen, auch wenn dies zur Folge hat, dass möglicherweise auch ein paar gesunde Menschen, fälschlicherweise als erkrankt klassifiziert werden. Wenn jedoch ein erkrankter Patient nicht erkannt wird, kann dies deutlich schwerwiegendere Folgen haben, als fälschlicherweise einen gesunden Menschen zu behandeln. [7]

K-Means Clustering

Funktionsweise

K-Means zählt zu den Clustering Verfahren, welche das Ziel verfolgen den Datensatz in Cluster genannte Gruppen zu unterteilen. Um diese Cluster zu finden, werden Mittelpunkte von Clustern gesucht, die bestimmte Regionen innerhalb der Daten repräsentieren. Bei der Suche nach diesen Mittelpunkten (Centruiden) werden anfangs N zufällige Mittelpunkte bestimmt. Danach wechselt der Algorithmus immer zwischen zwei Schritten hin und her:

1. Jeden Datenpunkt dem nächstgelegenen Cluster zuordnen (Abstände werden mit Euklidischer Distanz berechnet).
2. Jeden Clustermittelpunkt auf den Mittelwert der ihm zugeordneten Datenpunkte setzen. Die Mittelpunkte werden wie folgt berechnet: $\mu_{\text{neu}} = \frac{1}{|C_K|}\sum_{x \in C_K} x$ wobei $|C_K|$ die Anzahl der Vektoren im Cluster entspricht.

Der Lernalgorithmus ist beendet, sobald sich die Mittelpunkte nicht mehr verändern. [3]

Ziel des Lernalgorithmus ist es die Abweichung innerhalb des Clusters weitestgehend zu minimieren. $W(C_K)$ misst die gesamte Abweichung der Punkte im Cluster C_K um ihren gemeinsamen Mittelpunkt.

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_K) \right\}$$

$$W(C_K) = \frac{1}{|C_K|} \sum_{i,i' \in C_K} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_K|} \sum_{i,i' \in C_K} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

Dieses Maß der Abweichung ändert sich nicht mehr, sobald der Algorithmus die idealen Mittelpunkte gefunden hat. [8]

Anwendung auf Datensatz

In dieser Seminararbeit soll herausgefunden werden, ob ein Datenpunkt eine Anomalie ist oder nicht. Daher gibt es für diesen Fall genau zwei zu findende Cluster (Anomalie oder keine Anomalie).

Abbildung 6 zeigt die Segmente der Test-Daten (D-Test) der Zeitreihe mit einer Fenstergröße 3 und einer Verschiebung von 1 (wie im Beispiel oben) angeordnet in dreidimensionalen Graphen. Bei dem linken Graphen stellen die roten Punkte die

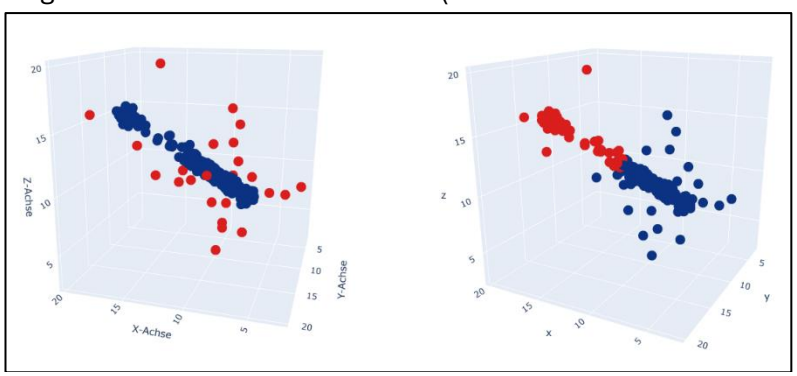

Abbildung 6: Beide Graphen zeigen die Visualisierung der Segmente, die zum Testen des Algorithmus verwendet wurden. Links = Soll-Zustand: Alle Segmente mit einer Anomalie sind rot markiert. Rechts = K-Means Clustering der Segmente

Segmente dar, die tatsächlich eine Anomalie beinhalten. Die blauen Datenpunkte sind Segmente ohne Anomalien.

Auf der rechten Seite ist die Vorhersage, die durch den K-Means-Cluster Algorithmus getroffen wurde. Hier ist zu erkennen, dass die vom Algorithmus vorhergesagten Label stark vom originalen linken Graphen abweichen. Diese Tatsache lässt vermuten, dass der Algorithmus mit dieser Fenstergröße keine guten Vorhersagen liefern kann.

Rücktransformation

Bevor wir den F1-Score auf die Vorhersagen des Cluster-Algorithmus anwenden können, müssen die Daten wie in Kapitel „Rücktransformation“ zurücktransformiert werden. Dies passiert auch mit dem Windowing-Verfahren.

Abbildung 7 verdeutlicht die schlechte Qualität der Vorhersage. Im linken Graphen ist die originale Einfärbung der Anomalien in den Testdaten abgebildet und auf der rechten Seite sind die rücktransformierten Werte des K-Means Cluster Algorithmus dargestellt. Vor allem der mittlere Bereich des rechten Graphen ist beinahe ausschließlich fälschlicherweise als Anomalie markiert.

Validierung

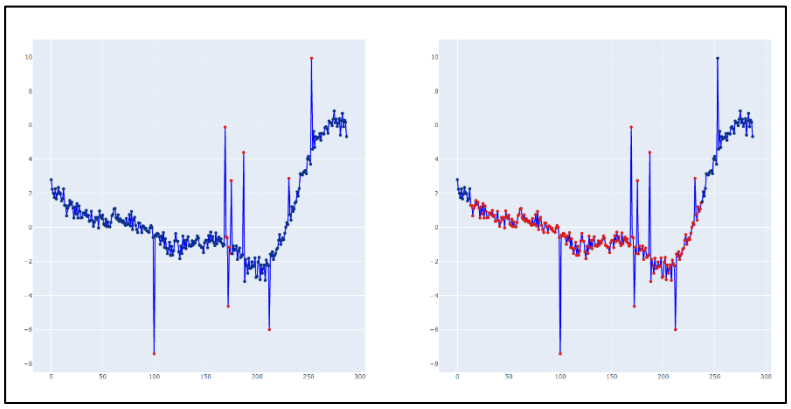

Abbildung 7:Rücktransformation der Segmente in Zeitreihe. Links das Original und rechts die Vorhersage mit K-Means Clustering

Bei der Validierung mit dem F1-Score werden die rücktransformierten Daten mit den originalen Daten verglichen. Hier wird unterschieden zwischen dem In-Sample Score, was dem F1-Score auf den Trainingsdaten entspricht und dem Out of Sample Score, der dem F1-Score des Algorithmus entspricht, wenn er auf die Testdaten angewendet wird.

Bei der Gittersuche wurde im Rahmen dieser Arbeit der Parameter der Window Size von 1 bis 9 mit Abstand 2 durchlaufen. Abbildung 8 zeigt den F1-Score für die verschiedenen Fenstergrößen wobei der rote Graph den In Sample Score und der blaue Graph den Out of Sample Score darstellt. Bei einer Fenstergröße von 1 hatte der In-Sample Score einen Wert von 0.11 und der Out of Sample Score lag bei 0.05. Dieses niedrige Ergebnis zeigt, dass keine Aussage getroffen werden kann. Mit steigender Fenstergröße nimmt auch der F1-Score ab. Was darauf schließen lässt, dass mit der Segmentierungsart und K-Means-Clustering hier keine Aussagen über Anomalien im Datensatz getroffen werden können.

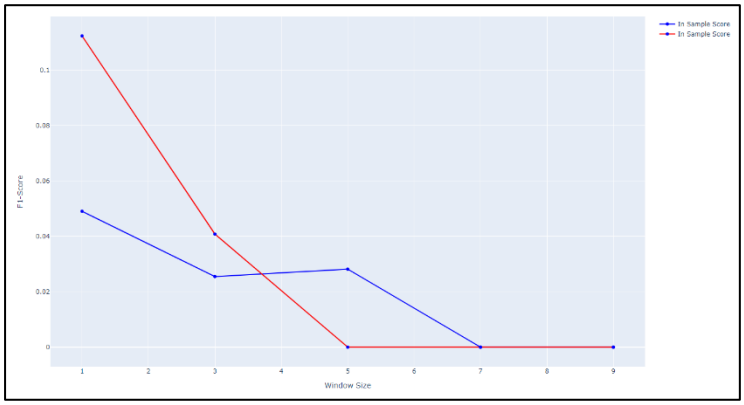

Abbildung 8: F1-Score pro Segmentgröße

Intuitiv hätte vermutet werden können, dass die Mittelpunkte der Cluster sich an bestimmten Kurvenformen (Segmenten) zentrieren, welche häufig im Datensatz vorkommen. Allerdings wurde in der Studie von E. Keogh und J. Lin, „Clustering of Time Series Subsequences is Meaningless“(2003) nachgewiesen, dass die Mittelpunkte einer Zeitreihe mit den Mittelpunkten von komplett zufällig generierten Segmenten nur vernachlässigbar voneinander abweichen. Dieses Ergebnis lässt darauf schließen, dass die Klassifizierung von segmentierten Zeitreihen mit Clustering Verfahren nicht zielführend ist. Der Grund für dieses Phänomen ist, dass die Clustermittelpunkte durch die Segmentierung sich in Sinus Kurven mit verschiedenen Verschiebungen anordnen. [6]

Fazit zu Clustering von Segmenten mit K-Means-Clustering

Die Segmentierung der Zeitreihe mit anschließender Klassifizierung durch den Lernalgorithmus K-Means Clustering hat basierend auf der Validierung durch den F1-Score keine aussagekräftige Klassifizierung von Anomalien im Datensatz treffen können. Daraus resultiert, dass dieses Verfahren mit diesem Datensatz nicht zielführend ist. Die weitere Recherche zeigte, dass dieses Problem nicht nur bei diesem Datensatz auftaucht, sondern an sich nicht funktional ist [6]. Dies liegt vor allem an dem Segmentierungsverfahren, welches die Informationen der Anomalien zur Klassifizierung durch den K-Means Clustering Algorithmus nicht sinnvoll transformiert.

One-Class SVM

One-Class SVM ist eine erweiterte Form des SVM-Lernalgorithmus, welche in erster Linie zur Erkennung von Anomalien, Ausreißern und Neuheiten in Datensätzen entwickelt wurde. One-Class SVM versucht dabei Instanzen zu finden, die erheblich von der Norm abweichen. Dadurch unterscheidet sich One-Class SVM von den traditionellen Klassifikationsalgorithmen wie K-Means, welche den Datensatz in Cluster einteilen will. Im Gegensatz zu dem herkömmlichen Support Vector Machine Verfahren braucht diese Unterkategorie keine markierten Daten, um das Modell zu trainieren. [9]

Mathematische Formulierung

Um den Datensatz in zwei Bereiche zu unterteilen wird eine **Hyperebene** (Hyperplane) konstruiert. Diese Hyperebene trennt die Ausreißer von den normalen Daten. Es gibt allerdings unendlich viele Geraden, die diese beiden Datensätze voneinander trennen können. SVM maximiert die Margin zwischen den Kategorien, um eine optimale Hyperebene zu finden. Dabei werden die Supportvektoren benutzt, welche am nächsten zur Hyperebene liegen. Supportvektoren sind ein wichtiger Bestandteil des Lernalgorithmus. Nur die Supportvektoren haben

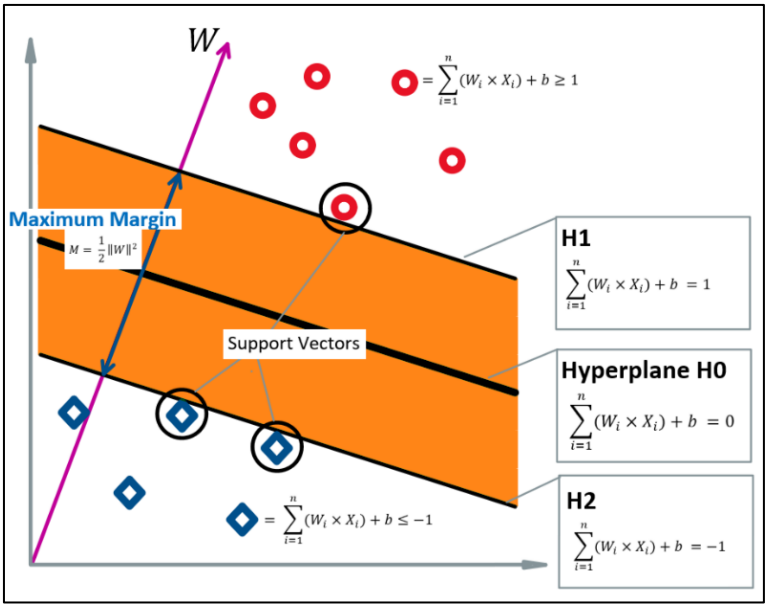

Abbildung 9: SVM-Beispielbild zur Erklärung von Grundbegriffen

einen Impact auf die Bestimmung der optimalen Hyperebene. [10]

Die Formel der Hyperebene lautet wie folgt:

$$H(W,X) = \sum_{i=1}^{n}(W_i X_i) + b \ = 0$$

Daraus folgen die Margin Grenzen H1mit $H(W,X) = 1$ und H2 mit $H(W,X) = -1$.Weiter lässt sich festhalten, dass alle Datenpunkte oberhalb der Hyperebene als positiv und alle unterhalb als negativ eingestuft werden können.

$$positiv\ (rot)\ f\ddot{u}r \sum_{i=1}^{n}(W_i X_i) + b \geq 1$$

$$negativ\ (blau)\ f\ddot{u}r = \sum_{i=1}^{n}(W_i X_i) + b \leq -1$$

Die maximale Margin kann wie folgt berechnet werden:

$$M = \frac{1}{2}\|W\|^2$$

Das Maximierungsproblem der Margin sowie die damit einhergehende optimale Hyperebene können daraufhin durch das Lagrange Verfahren unter der Nebenbedingung $H(W,X) \geq 1$ ermittelt werden.

$$L = \frac{1}{2}\left||W|\right|^2 - \sum_{i=1}^{n}\alpha_i[y_i(WX_i + b) - 1]$$

$$\frac{\partial L}{\partial W} = W - \sum_{i=1}^{n}\alpha_i y_i X_i = 0 \ \Rightarrow \ \boldsymbol{W = \sum_{i=1}^{n}\alpha_i y_i X_i}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = \ 0$$

$$\Rightarrow L = -\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i}\sum_{j}\alpha_i\alpha_j y_i y_j X_i X_j$$

Wenn der Datensatz nicht eindeutig trennbar ist, kann eine sogenannte „Slag-Variable“ (ζ_i) hinzugefügt werden. Diese Variable ermöglicht es mehr Spielraum für Outlier zu gewährleisten. Das daraus resultierende Verfahren wird auch als „Soft Margin SVM“ bezeichnet. Das neue y sieht dann wie folgt aus:

$$y_i(WX_i + b) \geq 1 - \ \zeta_i$$

$$mit\ (\zeta_i \geq 0, i = 1, \dots, m)$$

Das daraus resultierende Minimierungsproblem sieht dann wie folgt aus:

$$min\frac{1}{2}\left||W|\right|^2 + C\sum_{i=1}^{m}\zeta_i$$

Hierbei ist das C ein Parameter, welcher steuert wie stark die Outlier berücksichtigt werden sollen. Ein kleines C betont die Wichtigkeit von (ζ_i) und verkleinert tendenziell die Margin. Dieser Parameter kann noch weiter aufgeteilt werden, um somit das genaue Verhältnis von Outliern und Support Vektoren zu definieren:

$$min\frac{1}{2}||W||^2 + \frac{1}{vn}\sum_{i=1}^{m}\zeta_i$$

$v \in (0,1]$ und wird als „Nu“ bezeichnet. Bei der Anwendung auf den vorliegenden Datensatz wird dieser Parameter genauer untersucht und für den Datensatz angepasst. Nu bestimmt die Anzahl der Outlier und Supportvektoren ($v = 0.1$ entspricht 10% der Daten, die innerhalb der Margin liegen).

Kernel Trick

Wenn die Daten nicht linear separierbar sind, muss ein sogenannter „Kernel Trick“ angewandt werden. So lassen sich beispielsweise Kreise oder komplexere Formen im Datensatz finden. Bei diesem Trick wird im Speziellen die Lagrange Gleichung aus dem vorherigen Kapitel abgeändert.

$$L = -\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j \boldsymbol{k(X_iX_j)}$$

$\boldsymbol{k(X_iX_j)}$ kann dabei verschiedene Funktionen beinhalten. Ein Beispiel wäre hierfür die Quadrierung der einzelnen Faktoren $\boldsymbol{k(X_iX_j) = X_1^2 + X_2^2}$, was in einigen Fällen eine Separierung bereits ermöglichen könnte. Die bekanntesten Kernel sind dabei:

Polynomial Kernel: $\boldsymbol{k(X_iX_j) = (X_iX_jC)^d}$ mit C als Konstante und d als „Degree of freedom“

RBF-Kernel: $\boldsymbol{k(X_iX_j) = exp(-\gamma\left||X_i - X_j|\right|^2)}$ bei dem ein kleines $\boldsymbol{\gamma}$ das Modell linearer macht. Ein großes $\boldsymbol{\gamma}$ macht das Modell stärker abhängig von den Supportvektoren. [11, 12, 8, 10]

SVM vs One-Class SVM

Um die nicht klassifizierten Daten in 2 Kategorien einzuteilen, muss One-Class SVM (im Gegensatz zu dem normalen SVM-Verfahren) zuerst festlegen, was als Ausreißer und was als Norm klassifiziert wird. Anhand der gegebenen Parameter und der gewählten Kernel-Funktion werden die Daten transformiert und durch eine Entscheidungsfunktion

$$f(x) = w \cdot k(x) - \rho$$

interpretiert. Wobei ρ (rho) der Parameter ist, der die Entscheidungsgrenze festlegt und vom Kernel-Trick und dessen Hyperparameter sowie dem v (Nu) abhängt. Dabei gibt v den maximalen Anteil der Anomalien (Fehler) an, die innerhalb der Entscheidungsgrenze liegen dürfen. ρ wird während des Trainings so berechnet, dass dieser Anteil eingehalten wird.

Dabei gilt:

$$norm: f(x) \geq 0;$$

$$anomalie: f(x) < 0$$

Anwendung auf Datensatz

Angewandt auf den Datensatz wird der NU-One-Class SVM Lernalgorithmus mit einem „RBF-Kernel“. Dadurch ergeben sich die folgenden Parameter, die einzustellen sind:

Nu (ν): Anzahl der Outlier und Supportvektoren. ($\nu = 0.1$ entspricht 10%)

Gamma (γ): Bestimmt wie stark linear oder polymer der Lernalgorithmus ist.

Window Size: Ist die Segmentgröße, die bei der Transformation benutzt wird.

Nu vs Window Size (Gamma = 0,3):

Abbildung 10 zeigt die Gegenüberstellung von Fenstergröße und dem Nu-Parameter bei einem konstanten Gamma von 0,3. Der Graph Graph bildet den „In Sample Score“ ab. Da es bei großen Fenstergrößen zu erheblichen Informationsverlusten in der Transformierung kommt, behandelt diese Seminararbeit nur Fenster im Bereich 1-9. Der Nu-Parameter bewegt sich in einem Bereich zwischen 0,02 und 0,8.

In-Sample Score: Hat bei einem Nu von 0,22 und einer Fenstergröße von 5 ein Maximum von rund **F1= 0.8889**.

Basierend auf dieser Information wählen wir im folgenden Vergleich ein konstantes **Nu von 0,2**.

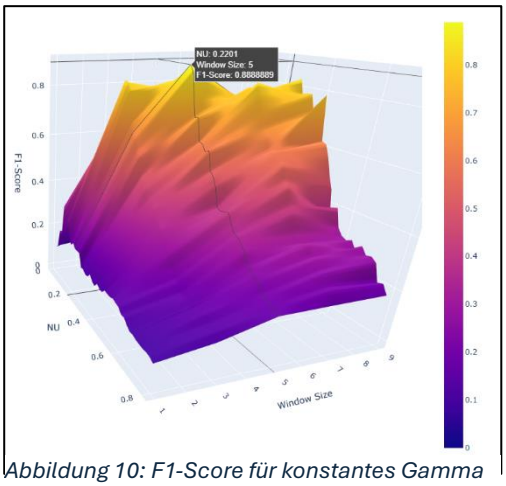

Abbildung 10: F1-Score für konstantes Gamma bei variierendem Nu von 0,01 bis 1 und variierender Fenstergröße von 1-9

Gamma vs Window-Size (Nu = 0,2):

Abbildung 11 zeigt die Gegenüberstellung von Gamma und der Window-Size bei einem konstanten Nu von 0.2.

In Sample Score: Erreicht sein Maximum bei einer Fenstergröße von 5 und einem Gamma von 0,4. Der **F1-Score** lieg hier bei rund **F1= 0,8889**

Daraus ergibt sich, dass der optimale Gamma Wert bei 0,5 liegen muss und da sowohl in der Gegenüberstellung von Abbildung 10 als auch in der Gegenüberstellung von Abbildung 11 die globalen Maxima bei einer Fenstergröße von 5 lagen, lässt sich daraus schlussfolgern, dass hier unser Optimum liegt.

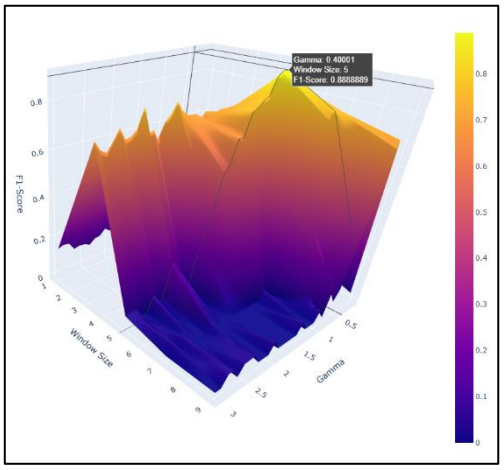

Abbildung 11: F1-Score bei konstantem NU=0.2 und variierendem Fenster zwischen 1-9 sowie variierendem Gamma zwischen 0.1-3.0

Anwendung auf Testdaten und Fazit

Abschließend werden die folgenden Hyperparameter aus dem vorherigen Abschnitt festgehalten und getestet:

Window Size = 5

Nu = 0,2

Gamma = 0,4

Mit diesen Werten erzielt der Lernalgorithmus auf den Trainingsdaten einen In Sample Score von rund **F1= 0,8889**. Auf den Testdaten kann mit diesen Werten allerdings nur ein F1-Score von rund **F1= 0,6316** erzielt werden.

Dieses Ergebnis deutet auf ein leichtes Overfitting auf den Trainingsdaten hin. Verändert man allerdings das Verhältnis von Test- und Trainingsdaten zu Gunsten der Testdaten ($D_{train} = 20\% => D_{train} = 40\%$) und validiert das Ergebnis mit denselben Hyperparametern, so ergibt sich hier ein In Sample Score von rund **F1= 0,8889** und ein besserer Out of Sample Score von **F1= 0,7895,** was ein deutlich besseres und ausgeglicheneres Ergebnis ist als zuvor.

Diese Tatsache lässt darauf schließen, dass der Test-Datensatz für eine aussagekräftige Bewertung zu klein gewählt war und der Algorithmus mit diesen Parametern durchaus fundierte Aussagen sowohl auf den Trainings- als auch auf den Testdaten treffen kann.

Fazit der Seminararbeit

Die Segmentierung von Zeitreihen durch das Windowing-Verfahren, sowie das anschließende Interpretieren durch den Cluster Algorithmus „K-Meanas Clustering“ führte durch die Natur des Clusterverfahrens, welches sich auf die Abstände der Datenpunkte fokussiert, nicht zu einem sinnvollen Ergebnis. Die Validierung der Daten ergab sowohl auf den Trainings- als auch auf den Test-Daten einen F1-Score unterhalb von **F1=0,15**. Daher ist diese Kombination von Verfahren für die Erkennung von Anomalien zwecklos.

Die Interpretation der Segmente durch den One Class SVM-Algorithmus erzielt mit einem In-Sample Score von rund **F1=0,8889** und einem Out of Sample Score von rund **F1=0,7895** (Bei einem D_{train} von 40%) deutlich bessere Ergebnisse als der K-Means Cluster Algorithmus. Für die Erkennung von Bot-Traffic oder Webseitenausfälle wäre diese Genauigkeit ausreichend und könnte hier zur Verwendung kommen. In kritischeren Bereichen wie der Medizin oder Raumfahrt wäre dieses Maß an Genauigkeit nicht ausreichend und müsste weiter optimiert werden.

Weitere Überlegungen zur Optimierung wären, die Genauigkeit durch ein effizienteres Transformierungsverfahren zu erhöhen, welches auch größere Segmente mit mehr Dimensionen zulässt. Ebenfalls müssten verschiedene Datensätze getestet werden, die verschiedene Herausforderungen im Bereich der Saisonalität sowie des Rauschens mit sich bringen.

Abschließend lässt sich festhalten, dass die Transformierung durch das Segmentierungsverfahren, die Interpretation durch den One Class SVM Algorithmus und die anschließenden Rücktransformierung mit dem oben genannten Datensatz zu einem fundierten Ergebnis gekommen ist. Um dieses Ergebnis für andere Datensätze zu verallgemeinern, sind jedoch weitere Tests mit weiteren Datensätzen nötig.

Verweise

[1] Wikipedia, „Geschichte der Schrift,“ 7 November 2024. [Online]. Available: https://de.wikipedia.org/wiki/Geschichte_der_Schrift.

[2] Elasticsearch , „Was bedeutet Anomalieerkennung?,“ 2024. [Online]. Available: https://www.elastic.co/de/what-is/anomaly-detection.

[3] A. C. M. &. S. Guido, Einführung inMachineLearningmit Python, Heidelberg: O’Reilly, 2017, pp. 123-.

[4] StatisticsEasily, „STATISTIK EINFACH LERNEN,“ 2022. [Online]. Available: https://de.statisticseasily.com/Glossar/Was-sind-saisonale-Schwankungen%3F/.

[5] datei.wiki, „Das statistische Rauschen verstehen,“ 2024. [Online].

Available: https://datei.wiki/definition/das-statistische-rauschen-verstehen/#:~:text=Definition%20von%20statistischem%20

Rauschen%3A%20Statistisches%20Rauschen%20bezieht%20sich,tats

%C3%A4chlichen%20Beziehungen%20in%20den%20Daten%20zu%20tun%20haben..

[6] E. Keogh und J. Lin, „Clustering of Time Series Subsequences is Meaningless,“ Computer Science & Engineering Department University of California - Riversid, Riverside, CA 92521, 2003.

[7] N. Lang, „Data Base Camp,“ 9 August 2023. [Online]. Available: https://databasecamp.de/statistik/f1-score.

[8] J. Gareth , D. Witten, T. Hastie, R. Tibshirani und J. Taylor, An Introduction to Statistical Learning, 2023.

[9] GeeksforGeeks, „geeksforgeeks.org,“ GeeksforGeeks, 2024. [Online]. Available: https://www.geeksforgeeks.org/understanding-one-class-support-vector-machines/. [Zugriff am 2024].

[10] Intuitive Machine Learning, Regisseur, *Support Vector Machines: All you need to know!*. [Film]. San Fransisco.2020.

[11] B. Schölkopf, A. Smola, R. Williamson und P. Bartlett, „New Support Vector Algorithms,“ 2000. [Online]. Available: https://www.stat.purdue.edu/~yuzhu/stat598m3/Papers/NewSVM.pdf. [Zugriff am 2024].

[12] scikit learn, „1.5.3. Online One-Class SVM,“ 2024. [Online]. Available: https://scikit-learn.org/stable/modules/sgd.html#online-one-class-svm. [Zugriff am 2024].