HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Lernen von maritimen Aktivitätsmustern aus Schiffsbewegungsdaten

Büşra Turuç

Bachelor-Arbeit im Studiengang „Angewandte Informatik“

25. September 2023

Autor Büşra Turuç
1476334
b.turuc@outlook.de

Erstprüfer: Prof. Dr. Ralf Bruns
Abteilung Informatik, Fakultät IV
Hochschule Hannover
ralf.bruns@hs-hannover.de

Zweitprüfer: Prof. Dr. Jürgen Dunkel
Abteilung Informatik, Fakultät IV
Hochschule Hannover
juergen.dunkel@hs-hannover.de

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die eingereichte Bachelor-Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Hannover, den 25. September 2023 Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In der heutigen Zeit werden ununterbrochen Daten generiert. Mit der zunehmenden Datengenerierung steigt auch die Relevanz der Datenerhebung und das Potenzial der Datenanalyse. Diese wachsende Datenflut ist auch im maritimen Sektor zu beobachten.

Maritime Traffic[1] sammelt mithilfe von 3000 Empfangsstationen täglich 520 Mio. Signale, die über Maritime Reporting Systems (MRS) von 180000 verschiedenen Schiffen übermittelt werden [mar]. Jedes Schiff, das einen Transceiver mitführt, versendet in regelmäßigen Abständen von zwei Sekunden bis drei Minuten Signale über Funk aus, die von Receivern an Land und anderen Schiffen im Sendebereich empfangen werden [AZ21].

Die daraus resultierende Datenflut bietet eine Reihe an Möglichkeiten, stellt jedoch auch eine Herausforderung dar, bei der manuelle Analysemethoden an ihre Grenzen stoßen. Es entsteht ein Bedarf an automatisierten Mechanismen, um die erhebliche Anzahl an Daten zu analysieren. Dementsprechend nimmt der Einsatz von Machine Learning und anderen Methoden an Relevanz zu. Machine Learning bezeichnet den Einsatz von Algorithmen, die es ermöglichen, aus Daten zu lernen, ohne dass diese explizit programmiert werden müssen. Durch die Analyse großer Datenmengen können Muster und Zusammenhänge erkannt werden, die bei manuellen Analysen schwer zu erfassen wären [ZL21].

Im maritimen Bereich hat eine solche Analyse ein großes Potenzial. Da Schiffe Informationen über ihre Position und Geschwindigkeit senden, können diese Meldungen genutzt werden, um den zunehmenden Schiffsverkehr zu überwachen. Dies könnte dazu beitragen, die Sicherheit durch Kollisionsvermeidung zu erhöhen, Rettungs- und Suchaktionen durchzuführen oder illegale Aktivitäten in Naturschutzgebieten aufzudecken [NAA+22].

Um dieses Potenzial auszuschöpfen, muss es möglich sein, aus den übermittelten Daten auf die aktuelle Aktivität eines Schiffes zu schließen. Domänenexperten müssen diese Aktivitätsmuster zuvor definieren und die Daten müssen mit Informationen aus verschiedenen Quellen anreichert werden, sodass aus einfachen Eigenschaften und Zuständen eines Schiffes auf seine Aktivität geschlossen werden kann [PAD+19].

[1] https://www.marinetraffic.com

1.1 Zielsetzung

Analysen über die Aktivitäten auf See wurden bereits mit verschiedenen komplexen Methoden erfolgreich durchgeführt, jedoch stellt sich die Frage, wie groß das Potenzial von Machine Learning Verfahren in der maritimen Domäne ist. Das Ziel dieser Arbeit ist es daher, die Frage zu beantworten, inwieweit maschinelle Lernverfahren in der Lage sind, Modelle zu lernen, die Aktivitätsmuster von Schiffen erkennen.

Die Ausgangssituation besteht in den bereits definierten Aktivitätsmuster, den Signalen, die ein Schiff in regelmäßigen Intervallen sendet, und zwei weiteren Datensätze, die punktuelle Ereignisse und durative Aktivitäten der Schiffe enthalten. Auf Grundlage dieser Daten soll abgeleitet werden, welches maritime Manöver ein Schiff zum Zeitpunkt der Übertragung durchführt. Dabei sollen alle Schritte bis zum gelernten Modell sowie die Bewertung der Ergebnisse beschrieben werden.

Da die Definitionen der Aktivitätsmuster vorliegen, kann aus ihnen abgeleitet werden, welche Aspekte von Bedeutung sind. Beispielsweise lautet die Definition für ein Schiff, welches momentan einer Rettungs- oder Suchmission nachgeht, wie folgt:

1. Bei dem Schiff handelt es sich um ein Rettungsschiff.
2. Das Schiff bewegt sich in einer Art, die einem Rettungsschiff gleicht.
3. Es hat eine Geschwindigkeit von mindestens 2.7 Knoten.
4. Die Bewegung und Geschwindigkeit wird über einen gewissen Zeitraum beibehalten.

In der Definition dieser beispielhaften Aktivität wird deutlich, dass Geschwindigkeit, Schiffstyp und Zeiträume alle von Bedeutung sein können [PAD+19].

Um das Ziel der Arbeit zu erreichen, müssen daher die folgenden Umstände beachtet werden:

1. **Integration verschiedener Datenquellen:** Da es sich bei den von Schiffen versendeten Signalen um simple Rohdaten handelt, müssen weitere Daten aus anderen Quellen integriert werden, um zusätzliche Informationen zu erhalten. Daher werden weitere Datensätze integriert, die in Kapitel *3 Maritimes Anwendungsszenario* näher beschrieben werden.
2. **Datenselektion:** Im maritimen Kontext werden immense Mengen an Daten generiert, die aufgrund ihrer Menge nicht im Ganzen verarbeitet werden können. Zudem sind je nach Aktivitätsmuster unterschiedliche Daten relevant. Daher muss die Definition einer Aktivität herangezogen werden und eine sinnvolle Selektion der Daten aus den verschiedenen Datensätzen stattfinden, um das Lernen zu ermöglichen.

Diese Selektion wird in Kapitel *4 Datenvorbereitung* beschrieben und je nach zu lernender Aktivität in Kapitel *5 Lernen der Modelle* durchgeführt.

3. **Analyse eines Datenstroms:** Bei den Daten handelt es sich um eine Aufzeichnung eines Datenstroms. Es sind mehrere Signale verschiedener Schiffe, die zu verschiedenen Zeitpunkten versendet wurden, enthalten. Diese Signale müssen in Verbindung gebracht werden, da sowohl der zeitliche Verlauf als auch die Zugehörigkeit eines Signals zu einem Schiff von Bedeutung ist [BD15]. Auf die Methodik, die in der Implementierung genutzt wird, wird in Abschnitt *2.2 Zeitreihendaten* eingegangen.

Werden diese Aspekte betrachtet, entsteht eine Grundlage, auf der Modelle gelernt und bewertet werden können. Für die Bewertung wird zu jedem gelernten Modell eine Übersicht erstellt, welche die Ergebnisse aufzeigt. In dieser Arbeit wird ein Modell, das in allen in Abschnitt *2.1.6 Modell Evaluation* erläuterten Messwerten einen Wert von mindestens 85% erreicht, als erfolgreich gelerntes Modell angesehen.

1.2 Vorgehensmodell

Die Vorgehensweise der Arbeit orientiert sich an dem 1996 entwickelten Cross Industry Standard Process For Data Mining (CRISP-DM). Das CRISP-DM Modell ist ein Referenzmodell, dass den Prozess in Data Science Projekten in sechs Phasen einteilt. Diese Phasen sollen im folgenden auf ihre relevanten Aspekte reduziert und in Bezug zu den angestrebten Zielen dieser Arbeit gesetzt werden.

1. **Business Understanding:** Das Ziel der ersten Phase ist es, die Geschäfts- und Data Mining Ziele zu definieren. Dazu muss die Domäne verstanden und relevantes Hintergrundwissen aufgebaut werden.
   Wie in Abschnitt *1.1 Zielsetzung* beschrieben ist das Hauptziel dieser Arbeit das Erlernen von Modellen, die in der Lage sind, maritime Aktivitätsmuster zu erkennen. Das nötige Hintergrundwissen bilden Konzepte des maschinellen Lernens, die in Kapitel *2 Grundlagen* beschrieben werden.

2. **Data Understanding:** In der zweiten Phase besteht das Ziel darin, ein Verständnis für die Daten aufzubauen. Daher soll in Kapitel *3 Maritimes Anwendungsszenario* eine Beschreibung der Datenerhebung und der relevanten Datensätze durchgeführt werden.

3. **Data Preparation:** In der dritten Phase, der Datenvorbereitung, werden alle Schritte durchlaufen, um aus den Rohdaten die Daten zu erzeugen, mit denen ein Machine Learning Algorithmus ein Modell erlernen kann. Die Phase beinhaltet

mehrere Schritte, die durchlaufen werden müssen, die in Kapitel *4 Datenvorbereitung* beschrieben werden.

4. **Modeling:** In der Modeling Phase werden die Algorithmen gewählt, ihre Parameter eingestellt und das Modell wird auf den Daten trainiert. Die Modeling Phase ist in Kapitel *5 Lernen der Modelle* beschrieben.
5. **Evaluation:** In der fünften Phase werden die gelernten Modelle auf ihre Qualität geprüft und es wird bewertet, ob die in der ersten Phase formulierten Ziele erfüllt werden konnten. Auch diese Phase wird in Kapitel *5 Lernen der Modelle* durchgeführt.
6. **Deployment:** In der letzten Phase sollen Erkenntnisse umgesetzt und ein Projektbericht erstellt werden. Die Phase besteht in dieser Arbeit aus einem abschließenden Fazit.

Die Phasen stellen keinen strikt sequentiellen Lebenszyklus eines Data Mining Projekts dar, da häufig das Zurückkehren zu einer vorherigen Projektphase nötig ist, um die gewünschten Ergebnisse zu erzielen. In dieser Arbeit werden sie jedoch sequentiell beschrieben [CCK+00].

1.3 Aufbau der Arbeit

Die Arbeit beginnt in Kapitel *2 Grundlagen* mit einer Einführung in die grundlegenden Konzepte des maschinellen Lernens und den Methoden der Leistungsbewertung von Modellen. Zusätzlich dazu werden Konzepte erläutert, die für die Arbeit mit Datenströmen relevant sind.
Daraufhin folgt ein Einblick in die maritime Domäne durch die Beschreibung der Datenerhebung und den verschiedenen Datensätzen (Kap. *3 Maritimes Anwendungsszenario*). Das 3. Kapitel beinhaltet ebenfalls die Beschreibung der verschiedenen Aktivitätsmuster. Darauf folgt die Vorbereitung der Daten (Kap. *4 Datenvorbereitung*). Hierbei werden alle Schritte aufgezeigt, um die Daten so aufzubereiten, dass Modelle gelernt werden können. In Kapitel *5 Lernen der Modelle* wird das Vorgehen beim Lernen jedes Aktivitätsmusters beschrieben und die Ergebnisse präsentiert sowie evaluiert. Abschließend folgt ein Fazit, das die Ergebnisse zusammenfasst und alternative Herangehensweisen beschreibt (Kap. *6 Fazit*).

2 Grundlagen

Gemäß der Phase *Business Understanding* muss zunächst relevantes Hintergrundwissen aufgebaut werden. Dazu werden in diesem Kapitel grundlegende Machine Learning Konzepte mit Bezug auf die maritime Domäne erläutert. Daraufhin soll eine Möglichkeit der Verarbeitung von Datenströmen im Machine Learning, die Zeitreihenanalyse, erläutert werden. Auch hierfür wird ein Beispiel aus dem Anwendungsszenario herangezogen.

2.1 Machine Learning

Machine Learning ist ein Teilgebiet der künstlichen Intelligenz. Es beschreibt die Fähigkeit eines Systems, aus Daten Wissen zu extrahieren. Ein Machine Learning Algorithmus lernt aus gegebenen problemspezifischen Daten bestimmte Muster, Gesetzmäßigkeiten oder Regeln, genannt Modell. Mit dem erlernten Modell kann er auf neuen Daten arbeiten.

Machine Learning hat das Ziel, die Fähigkeit eines Menschen nachzubilden, aus seinen Erfahrungen heraus zu lernen, Muster und charakterisierende Eigenschaften eines Objekts zu erkennen und dadurch Fragen zu beantworten oder Probleme zu lösen. Diese Fähigkeit wird auf der Grundlage von vorherigen Betrachtungen gebildet und ist die Kernidee des maschinellen Lernens [ZL21].

2.1.1 Terminologie

Zunächst soll als Grundlage für alle folgenden Konzepte die gängige Machine Learning Terminologie definiert werden. Insbesondere die Begriffe Sample, Feature und Label sind im maschinellen Lernen von Bedeutung.

Sample Die Grundlage in jedem Machine Learning Vorhaben bilden Daten. Um aus realen Objekten oder Konzepten Informationen zu extrahieren, muss eine Beschreibung dieser Objekte in Form von strukturierten Daten vorliegen. Eine Beschreibung einer

einzelnen betrachteten Entität wird allgemein als Datenpunkt bezeichnet. Im Machine Learning werden diese Datenpunkte Sample genannt. Eine Menge an gesammelten Sample bilden einen Datensatz, der in Form einer Tabelle dargestellt wird.

#	sourcemmsi	speed	shiptype	t
1	124563555	9.0	Trawler	1692094080
2	450239457	5.0	Sailboat	1692094120
...	...	...	...	...
99	450239457	7.0	Sailboat	1692094220
100	124563555	6.0	Trawler	1692094225

Tabelle 2.1: Fiktiver Beispieldatensatz

Im Beispieldatensatz 2.1 wurden 100 Sample in einen Datensatz zusammengefasst. Die betrachteten realen Entitäten sind Signale verschiedener Schiffe und ein Sample ist die repräsentative Darstellung dieses Signals, welches jeweils in einer Zeile der Tabelle steht.

Feature und Werte Die einzelnen Attribute, die ein Sample beschreiben, werden als Feature bezeichnet. Feature sind eine messbare Eigenschaft oder ein Attribut eines Samples. In dem Datensatz 2.1 sind die Features, die das Signal ausmachen, die Identifikationsnummer des Schiffs *sourcemmsi*, welches das Signal versendet hat, die Geschwindigkeit *speed*, der Typ des Schiffs *shiptype* und ein Zeitstempel *t*, der angibt, wann das Signal versendet wurde.

Die Features nehmen Werte an, die eine konkrete Messung oder Beobachtung dieses Features sind. Bei dem ersten Sample nimmt das Feature *sourcemmsi* den Wert *1692094080*, *speed* den Wert *9.0* und *shiptype* den Wert *Trawler* an. Laut dem Wert des Features *t* wurde das Signal zum Zeitpunkt *1692094080* erstellt.
Je nach Feature ist der Typ sowie die Einheit, die ein Wert annehmen kann, unterschiedlich. Die Identifikationsnummer und die Geschwindigkeit nehmen numerische Werte an, der Schiffstyp ist ein kategorisches Feature und der Zeitstempel ist ein zeitlicher Wert in Form eines Unix-Zeitstempels.

Label Daten können in zwei Arten aufgeteilt werden, beschriftete (*labeled*) und unbeschriftete (*unlabeled*) Daten. Unbeschriftete Daten sind eine einfache Sammlung an Samplen. Bei beschrifteten Daten hingegen wird jedem Sample ein Label zugeschrieben. Daten zu labeln bedeutet, diese Einträge hinzuzufügen.
Wenn die Geschwindigkeit mehrerer Fischereifahrzeuge gesammelt wird, handelt es sich um unbeschriftete Daten (Tab. 2.2). Wenn den Samples des Datensatzes zugeordnet wird, ob die aufgezeichnete Geschwindigkeit für ein Fischereifahrzeug als überdurchschnittlich

schnell (*above*), durchschnittlich (*normal*) oder langsam (*below*) gilt, handelt es sich um beschriftete Daten (Tab. 2.3).

shiptype	**speed**
Trawler	9.0
Trawler	7.5
Trawler	15.4
Trawler	6.3
Trawler	14.9

Tabelle 2.2: Unbeschriftete Daten

shiptype	**speed**	**movingSpeed**
Trawler	9.0	normal
Trawler	7.5	below
Trawler	15.4	above
Trawler	6.3	below
Trawler	14.9	above

Tabelle 2.3: Beschriftete Daten

In diesem Beispiel kategorisiert das Label *movingSpeed* die einzelnen Datenpunkte nach der aufgezeichneten Geschwindigkeit und weist jedem Sample einen der Werte *above*, *normal* oder *below* zu.

Allgemein ist ein beschrifteter Datensatz wie in der Tabelle 2.4 aufgebaut.

$Sample\, x \in X$	$feaure_1$	$feature_2$	...	$feature_n$	$Label\, y \in Y$
x_1					y_1
...					...
x_m					y_m

Tabelle 2.4: Der allgemeine Aufbau eines beschrifteten Datensatzes

Die Menge aller Sample X bilden den Datensatz 2.4, welcher aus m Samplen besteht. Jedes $x_i \in X$ ist ein Sample im Datensatz und enthält die Beschreibung einer Entität in Form von n verschiedenen Features und ihren Werten. Der Wert y ist das Label und Y die Menge aller möglichen Label, auch Labelraum genannt, die einem Sample zugeordnet werden können. Ein gelabeltes Sample x_i kann daher als (x_i, y_i) geschrieben werden, wobei y_i das Label des Samples x_i ist [ZL21].

2.1.2 Supervised Learning

Machine Learning wird in zwei Typen unterteilt, dem überwachtem Lernen (*Supervised Learning*) und dem unüberwachten Lernen (*Unsupervised Learning*). Beim Unsupervised Learning werden unbeschriftete Daten genutzt, beim Supervised Learning hingegen werden beschriftete Daten benötigt. Der Unterschied besteht also darin, dass dem Algorithmus ein Datensatz aus gelabelten Samplen gegeben wird.

Beim Supervised Learning wird anhand des Datensatzes, der alle Sample und die zugehörigen Label enthält, ein Modell gelernt, das beim Anwenden auf neue, nicht gelabelte Daten jedem Sample x_i ein Label $y \in Y$ zuordnet. Ist ein Datensatz gegeben, muss dieser für den Algorithmus vorbereitet werden. Diese Vorbereitung soll beispielhaft am Datensatz 2.5 aufgezeigt werden.

#	**sourcemmsi**	**speed**	**shiptype**	**timestamp**	**movingSpeed**
1	124563555	9.0	Trawler	1692094080	normal
2	450239457	5.0	Trawler	1692094120	below
3	124563555	12.0	Trawler	1692094150	above
4	450232257	6.8	Trawler	1692094180	below
5	450239457	8.7	Trawler	1692094220	normal
6	124563555	6.0	Trawler	1692094225	below

Tabelle 2.5: Fiktiver Beschrifteter Beispieldatensatz

Zunächst wird der Datensatz aufgeteilt in X und y. Wobei X alle Sample enthält und y die zugehörigen Label.

$$X = \begin{bmatrix} 124563555 & 9.0 & \text{Trawler} & 1692094080 \\ 450239457 & 5.0 & \text{Trawler} & 1692094120 \\ 124563555 & 12.0 & \text{Trawler} & 1692094150 \\ 450232257 & 6.8 & \text{Trawler} & 1692094180 \\ 450239457 & 8.7 & \text{Trawler} & 1692094220 \\ 124563555 & 6.0 & \text{Trawler} & 1692094225 \end{bmatrix} \qquad y = \begin{bmatrix} \text{normal} \\ \text{below} \\ \text{above} \\ \text{below} \\ \text{normal} \\ \text{below} \end{bmatrix}$$

Anhand y wird auf den Labelraum Y geschlossen, welcher alle möglichen Label beinhaltet.

$$Y = \begin{bmatrix} \text{normal} \\ \text{below} \\ \text{above} \end{bmatrix}$$

Daraufhin soll der Algorithmus eine Funktion f finden, die jedem Sample x_i ein Label aus dem Labelraum Y zuordnet.

$$f : x \to Y \tag{2.1}$$

In der Praxis wird dafür der gesamte Datensatz aufgeteilt in Trainingsdaten und Testdaten. Durch das Aufteilen entstehen X_{train}, X_{test}, y_{train} und y_{test}. Die Sample in X_{train} und ihre Label in y_{train} werden dann genutzt, um den Machine Learning Algorithmus zu trainieren und ein Modell zu lernen. Wurde das Modell gelernt, wird es genutzt, um die Sample in X_{test} zu labeln. Die vorhergesagten Label des Modells werden mit den tatsächlichen Label in y_{test} verglichen, um die Leistung des Modells zu überprüfen. Ein funktionales Modell, das viele der Testdaten korrekt gelabelt hat, kann dann auf neue, nicht gelabelte Daten angewendet werden.

Die Aufgabe des überwachten Lernens besteht demnach darin, eine Verbindung zwischen spezifischen Datenmerkmalen und den zugehörigen Labeln zu erkennen [ZL21].

In dieser Arbeit wird ausschließlich mit Supervised Learning Algorithmen gearbeitet. Beim Supervised Learning kann die Leistung des Modells objektiv anhand von Messwerten, die in Abschnitt *2.1.6 Modell Evaluation* beschrieben sind, bewertet und geprüft werden, ob das Modell den gewünschten Qualitätsstandard erfüllt. Durch die Abhängigkeit von gelabelten Daten müssen jedoch alle genutzten Datensätze gelabelt sein oder gelabelt werden.

2.1.3 Klassifikation

Das überwachte Lernen ist unterteilt in Klassifikations- und Regressionsprobleme. Bei beiden Arten ist das Ziel wie zuvor erwähnt eine Funktion f zu finden, die einem Sample x ein Label aus dem Labelraum Y zuordnet. Der Unterschied zwischen Regression und Klassifikation besteht darin, aus welchen Werten Y besteht. Bei Regression handelt es sich bei Y um kontinuierliche numerische Werte. Bei der Klassifikation hingegen handelt es sich um eine diskrete Menge. Klassifikation beschäftigt sich mit der Zuordnung von Samplen zu vordefinierten Labeln, auch Klassen genannt. Daher wird die Funktion f, bei Klassifikationsproblemen, als Klassifikationsfunktion bezeichnet und die Machine Learning Algorithmen für Klassifikationsprobleme als Classifier (Klassifikator) [MP18].

In der maritimen Domäne sind die Aktivitätsmuster kategorische Label, daher besteht Y aus diskreten Werten und es wird mit Klassifikationsalgorithmen gearbeitet.
Für ein besseres Verständnis sollen einige der gängigsten Algorithmen kurz vorgestellt werden.

K-Nearest Neighbors (k-NN)

Der k-Nearest Neighbor Algorithmus ist einer der einfachsten Machine Learning Algorithmen, der auf der Annahme basiert, dass Sample, die laut den Werten ihrer Features „nah“ beieinanderliegen, zu der gleichen Klasse gehören.

Die Funktionsweise des k-NN Classifiers kann einfach visualisiert werden. Um die Erklärung verständlich zu machen, wird hier ein Datensatz genutzt, in dem nur zwei numerische Features vorhanden sind und nur zwei verschiedene Klassen, die positive oder die negative Klasse, zu denen ein Sample gehören kann. Eine solche Klassifizierung wird als binäre Klassifizierung bezeichnet. Da es nur zwei Features gibt, können die Sample des Trainingsdatensatzes wie in Abbildung 2.1[1] in einem zweidimensionalen Raum eingetragen werden, der von den Features aufgespannt wird (Feature space).

[1] Erstellt mit https://matplotlib.org/

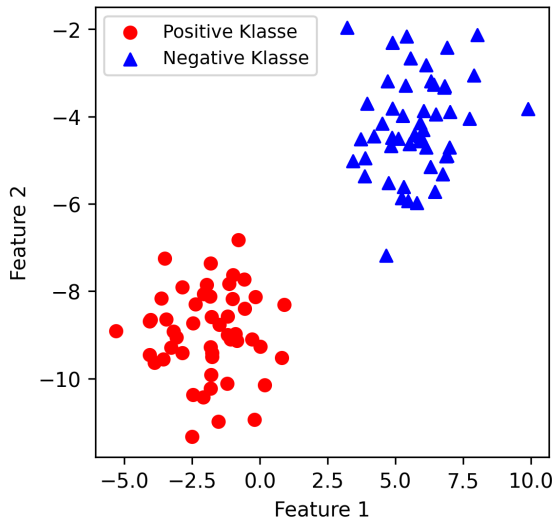

Abbildung 2.1: Sample eines Datensatzes

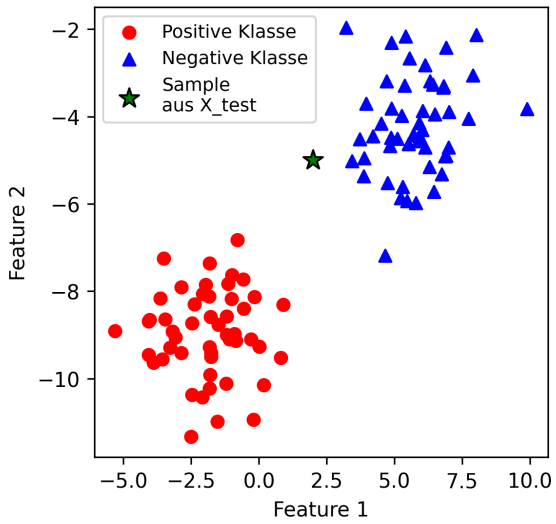

Abbildung 2.2: Klassifikation in k-NN

Die Klassen sind in den Abbildungen 2.1 und 2.2[2] durch ihre Farbe und Form gekennzeichnet. Die Sample, die zu der positiven Klasse gehören, sind als rote Punkte dargestellt. Die Sample der negativen Klasse hingegen als blaue Dreiecke. Der Algorithmus merkt sich die Sample des Trainingsdatensatzes aus der Abbildung 2.1 und klassifiziert anhand der Trainingssample die Testsample.
Der Parameter k beschreibt die Anzahl der „Nachbarn“, die betrachtet werden sollen bei der Klassifizierung. Wird $k = 1$ gesetzt, wird für jedes Sample aus dem Testdatensatz mithilfe einer Distanzfunktion sein Nachbar bestimmt, der nach dem genutzten Distanzmaß am nächsten liegt. Dem Testdatenpunkt wird dann die Klasse zugeordnet, die sein Nachbar hat. Wird $k = 2$ gesetzt, wird der neue Datenpunkt in der Abbildung 2.2, welcher als Stern gekennzeichnet ist, als negative Klasse klassifiziert, da seine nächsten zwei Nachbarn zu der negativen Klasse gehören. Wenn mehrere Nachbarn betrachtet werden, wird die Klasse gewählt, die unter den Nachbarn am häufigsten vorkommt. Bei einer ausgeglichenen Anzahl von Klassen unter den Nachbarn werden weitere Berechnungen durchgeführt, um eine endgültige Klassifikation durchzuführen [Kub21].

Random Forest

Um das Konzept des Random Forest Classifiers zu veranschaulichen, wird zunächst erklärt, was ein Decision Tree ist. Bei dem Decision Tree Classifier handelt es sich um einen Klassifizierungsalgorithmus, der auf Grundlage des Datensatzes rekursiv einen Entscheidungsbaum erstellt, der dann genutzt wird, um neue Sample zu klassifizieren. Um einen

[2] Erstellt mit https://matplotlib.org/

Baum zu erstellen, durchsucht der Algorithmus alle Sample und findet die Features, die am meisten über die Klassen aussagen. Ein Baum besteht aus einer Abfolge von Wenn/Dann-Fragen, wobei jeder Knoten eine Frage darstellt. Dieser rekursive Prozess ergibt einen Binärbaum von Entscheidungen. Ein solcher Baum ist in 2.3 dargestellt. In diesem Beispiel wird einem Schiff anhand seines Schiffstypen und anhand der Geschwindigkeit eine der Klassen below, normal oder above zugeordnet.

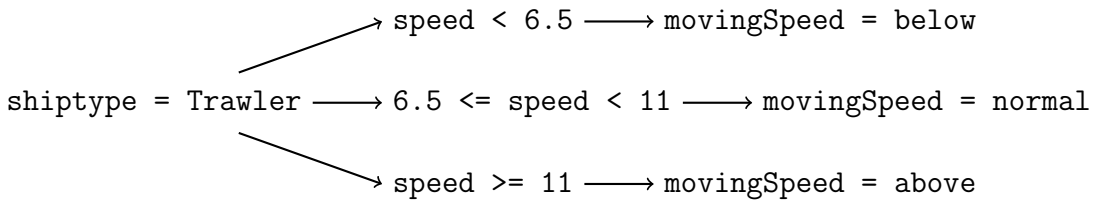

Abbildung 2.3: Beispielhafter Decision Tree

Auf Grundlage dieses Prinzip arbeitet der Random Forest Classifier. Es handelt sich hierbei um einen *Esamble Classifier*, welcher mehrere Classifier vereint, um bessere Ergebnisse zu erzielen. Es wird eine Vielzahl an Entscheidungsbäumen auf Teilmengen der Testdaten erstellt. Während jeder einzelne Decision Tree im Random Forest eine Gruppe von Entscheidungsregeln erstellt, erfolgt die endgültige Klassifikation durch eine Abstimmung über die Ergebnisse aller erstellten Bäume [ZL21].

Support Vector (SVC)

Der Support Vector Classifier ist ein binärer Classifier, das bedeutet, er funktioniert ausschließlich für binäre Klassifizierung. Das Ziel des Classifiers ist es, die zwei Klassen im Raum eindeutig zu trennen. Die Abbildung 2.4[3] visualisiert hierbei das Vorgehen.

[3]Erstellt mit https://matplotlib.org

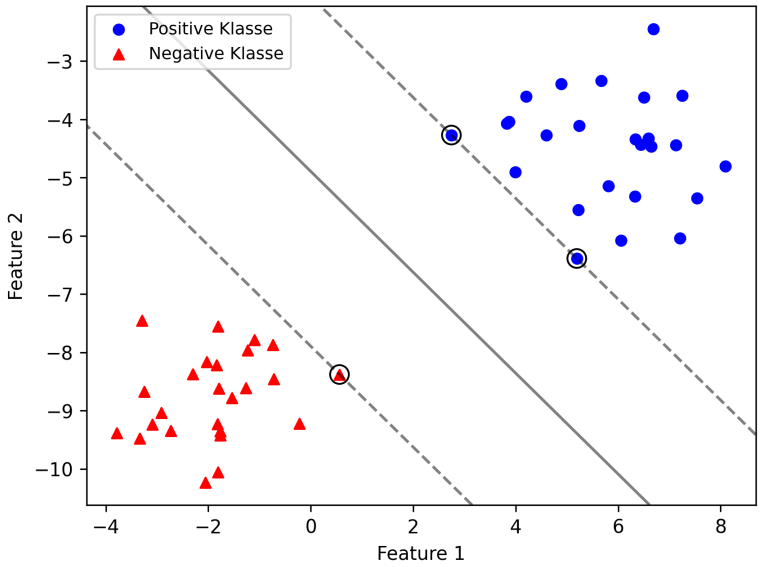

Abbildung 2.4: Support Vector Classifier

Die Sample der positiven Klasse wurden als blaue Punkte und die Sample der negativen Klasse als rote Dreiecke gekennzeichnet. Die mittlere Gerade ist die sogenannte *Hyperplane* (Hyperebene), die vom SV Classifier gefunden wurde und die zwei Klassen voneinander trennt. Die Geraden auf der rechten und linken Seite des Hyperplanes sind die *Margins*. Je nachdem, auf welcher Seite der Hyperplane, ein neuer Datenpunkt, liegt, wird er klassifiziert. Der Classifier versucht nicht nur, die Datenpunkte zu trennen, um neue Datenpunkte korrekt zu klassifizieren, sondern auch sicherzustellen, dass die Trennung der zwei Klassen mit der größten möglichen Margin stattfindet. Diese größte mögliche Entfernung wird mithilfe von den *Support Vectors* berechnet. Support Vectors sind die Datenpunkte, die am nächsten zum Hyperplane liegen. Diese sind in der Abbildung 2.4 die eingekreisten Datenpunkte [Kub21].

2.1.4 Mehrklassen-Klassifikation

Mehrklassen-Klassifikation bezeichnet Klassifikationsaufgaben, bei denen die Anzahl der Klassen, denen ein Sample zugehörig sein kann, größer als zwei ist. Viele Machine Learning Algorithmen wie k-NN, Decision Tree oder Random Forest können direkt für Mehrklassen-Klassifikation verwendet werden, ohne dass Modifikationen erforderlich sind. Andere wiederum, wie der Support Vector Classifier, sind für binäre Klassifikation ausgelegt. Die Machine Learning Algorithmen, die für binäre Klassifikation ausgelegt sind, können angepasst werden, um Mehrklassen-Klassifikation durchzuführen. Eine gängigere Methode ist es jedoch, sie in ihrer eigentlichen Form zu verwenden und aus

dem Problem eine binäre Klassifikation zu machen, indem der Datensatz angepasst wird. Daher müssen beim Nutzen vieler Algorithmen die Klassen angepasst werden.

Die grundlegende Idee bei der Umwandlung eines Mehrklassen-Klassifikationsproblems in eine binäre Klassifikation ist es, den Datensatz zu zerlegen. Für die Zerlegung gibt es mehrere Methoden. Eine dieser Methoden ist die *One-vs-Rest (OvR)* Methode, welche im Folgenden beschrieben werden soll.

Gibt es n verschiedene Klassen, wird für jede Klasse eine Zerlegung durchgeführt, die die jeweilige Klasse als die positive Klasse ansieht und die restlichen Klassen in die negative Klasse zusammenfasst.

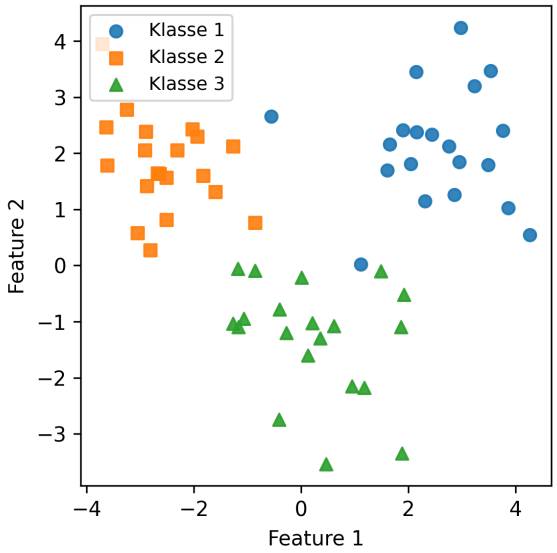

Abbildung 2.5: Mehrklassen – Klassifizierung

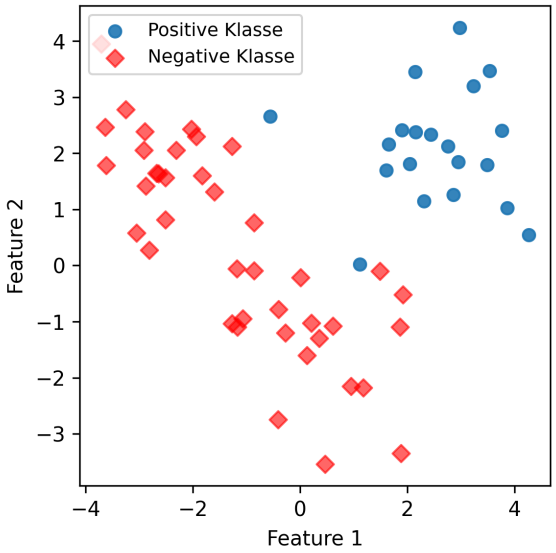

Abbildung 2.6: One vs. Rest – Klassifizierung

In der Abbildung 2.5[4] ist der gesamte Trainingsdatensatz abgebildet. Innerhalb des Datensatzes gibt es drei Klassen, deren Sample jeweils durch unterschiedliche Formen gekennzeichnet werden. Soll nun ein Modell gelernt werden, werden bei der OvR-Methode drei Classifier trainiert. Dabei wird jede Klasse nacheinander als positiv betrachtet, während die restlichen Klassen als eine negative Klasse betrachtet werden. Diese Methode ist in der Abbildung 2.6[5] veranschaulicht. Der erste Classifier betrachtet Klasse 1 als die positive Klasse, hier als Kreise abgebildet. Die anderen Klassen werden in eine Klasse, der negativen, zusammengefasst. Auf dem nun binär klassifizierten Datensatz wird ein Modell gelernt. Dieser Prozess wird für alle Klassen durchgeführt und es werden insgesamt drei Modelle gelernt.

[4]Erstellt mit https://matplotlib.org
[5]Erstellt mit https://matplotlib.org

Daraufhin werden neue Sample klassifiziert. Wenn während des Klassifizierens neuer Sample nur einer der Modelle das Sample als positiv markiert, ist dies die endgültige Klassifikation. Wenn jedoch mehrere Classifier die neue Probe als positiv markieren, werden weitere Berechnungen durchgeführt, um zu entscheiden, welche Klassifikation gewählt wird [ZL21].

2.1.5 Over- und Underfitting

Ein Modell, das gute Vorhersagen trifft, daher Sample korrekt klassifiziert, die neu für das Modell sind, hat eine gute Generalisierungsfähigkeit. *Overfitting* (Überanpassung) und *Underfitting* (Unteranpassung) sind Konzepte, die beschreiben, wie komplex ein Modell ist. Die Komplexität eines Modells hat einen direkten Einfluss darauf, wie gut es generalisiert.

Overfitting Overfitting bedeutet, dass ein Modell stark an den Trainingsdatensatz angepasst und daher zu komplex ist. Das Modell verliert die Fähigkeit, auf unbekannten Datensätzen zu generalisieren. Dies führt zu einer schlechten Leistung bei der Anwendung auf neue Daten.

Underfitting Underfitting tritt auf, wenn ein Modell zu einfach ist, um die zugrunde liegenden Muster in den Trainingsdaten zu erfassen. Das Modell ist nicht in der Lage, neue Sample korrekt zu klassifizieren, da es die Strukturen der Daten nicht angemessen berücksichtigt. Das Ergebnis ist ein Modell, das eine schlechte Leistung bei der Verwendung auf den Testdaten zeigt.

In beiden Fällen ist das Modell nicht in der Lage zu generalisieren. Das Ziel beim Trainieren eines Modells besteht darin, ein Gleichgewicht zu finden, bei dem das Modell in der Lage ist, die wichtigen Muster in den Daten zu erfassen, ohne sich zu stark an die Trainingsdaten anzupassen. Um Over- und Underfitting zu vermeiden, werden die folgenden Punkte betrachtet:

1. **Auswahl der Features:** Die Features, die in einem Datensatz erhalten bleiben sollen, müssen richtig gewählt werden. Ist dies nicht der Fall, kann das Modell auch bei einfachen Aktivitätsmustern komplexe Gesetzmäßigkeiten erlernen.
2. **Größe des Trainingsdatensatzes:** Der Trainingsdatensatz muss ausreichend groß sein, damit Gesetzmäßigkeiten im Datensatz erkannt werden können.
3. **Modellauswahl und Parametereinstellung:** Je komplexer der Machine Learning Algorithmus ist, desto komplexer kann auch das Modell werden. Daher müssen

geeignete Algorithmen gewählt und ihre Parameter, je nach Anforderungen, angepasst werden.

2.1.6 Modell Evaluation

Wie bereits in Abschnitt *2.1.2 Supervised Learning* erwähnt, gibt es eine Vielzahl an Möglichkeiten, die Leistung eines Modells zu evaluieren. Im Folgenden wird eine Übersicht über die Metriken, welche zu jedem gelernten Modell berechnet werden, beschrieben.

Confusion Matrix Als Grundlage für alle Messwerte dient die Confusion Matrix. Die Confusion Matrix zeigt an, wie viele Sample korrekt und wie viele inkorrekt klassifiziert wurden.
Ist eine Klassifikation mit nur zwei Klassen, der positiven Klasse A und der negativen Klasse B gegeben, ist die Confusion Matrix wie folgt aufgebaut:

	Zugewiesene Klasse A	Zugewiesene Klasse B
Tatsächliche Klasse A	TP	FN
Tatsächliche Klasse B	FP	TN

Tabelle 2.6: Allgemeine Confusion Matrix

Die Confusion Matrix bei einer binären Klassifikation stellt die folgenden Fälle dar:

True Positives (TP): Anzahl der Sample, die zu der positiven Klasse gehören und auch als solche klassifiziert wurden.

True Negatives (TN): Anzahl der Sample, die zu der negativen Klasse gehören und auch als solche klassifiziert wurden.

False Positives (FP): Anzahl der Sample der negativen Klasse, die als positiv klassifiziert wurden.

False Negatives (FN): Anzahl der Sample der positiven Klasse, die als negativ klassifiziert wurden.

	Zugewiesene Klasse A	Zugewiesene Klasse B
Tatsächliche Klasse A	50	5
Tatsächliche Klasse B	3	40

Tabelle 2.7: Konkrete Confusion Matrix

In der Confusion Matrix 2.7 wurden bei insgesamt 55 Samplen der Klasse A, 50 korrekt als A klassifiziert und 5 fälschlicherweise als Klasse B.

Accuracy Der Accuracy Wert gibt an, wie hoch der Prozentsatz an korrekt gelabelten Samplen ist.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

Im aufgeführten Beispiel 2.7 liegt der Accuracy Wert bei 91,8%, da bei 98 Samplen für 90 die korrekte Klasse bestimmt wurde. Bei einem Datensatz, in dem die Klassen stark in ihrer Größe abweichen, kann der Wert irreführend sein. Ein Modell kann bei stark unausgeglichenen Klassengrößen hohe Accuracy Werte erreichen, wenn nur die überwiegende Klasse vorhergesagt wird. Gäbe es nur 3 Sample der Klasse B und 97 Sample der Klasse A, wäre die Accuracy eines Modells, das jedes Sample als A klassifiziert, bei 97% und das Modell wäre nicht in der Lage, selten auftretende Klassen zu erkennen. Deshalb werden weitere Messwerte benötigt, die vor allem bei nicht ausbalancierten Klassen sinnvoll sind.

Precision Precision gibt das Verhältnis der korrekt positiven (TP) Fälle zur Gesamtanzahl der positiven Klassifikationen an. Eine hohe Precision sagt aus, dass die meisten vom Modell als positiv klassifizierten Sample tatsächlich positiv sind. Eine niedrige Precision zeigt, dass das Modell viele falsch positive Klassifikationen durchgeführt hat. Der Wert gibt an, wie wahrscheinlich es ist, dass der Classifier richtig liegt, wenn er ein Beispiel als positiv kennzeichnet.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

In der Übersicht über die Metriken wird die Macro Average Precision, also die durchschnittliche Precision über alle Klassen hinweg berechnet. Hierbei wird jeweils eine Klasse als Positive Klasse gesehen, die Precision wird berechnet und der Durchschnitt aller Precision Werte wird berechnet. Da jede Klasse bei der Berechnung gleich gewichtet wird, beeinflusst die Anzahl der Sample jeder Klasse nicht das Ergebnis.

Recall Recall berechnet das Verhältnis der korrekt positiven (TP) Fälle zur Gesamtanzahl der tatsächlich vorhandenen positiven Fälle. Der Wert gibt an, wie viele Sample der positiven Klasse korrekt vom Modell erkannt wurden. Ein hoher Recall zeigt, dass das Modell die meisten der tatsächlich positiven Sample korrekt klassifiziert hat. Ein niedriger Recall hingegen deutet darauf hin, dass viele positive Sample inkorrekt klassifiziert wurden.

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Bei der Evaluation eines Modells wird analog zur Precision der Macro Average Recall betrachtet.

F_1-Score Der F1-Score ist das harmonische Mittel zwischen Precision und Recall. Ein hoher F1-Score zeigt an, dass sowohl die Anzahl der falsch positiven als auch der falsch negativen Klassifikationen gering ist.

$$F_1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.5}$$

Auch beim F_1-Score wird der Macro Average berechnet. Ein niedriger Macro Average F_1-Score sagt aus, dass das Modell bei der Klassifizierung viele Fehler macht und Anpassungen nötig sind [Kub21].

Wie bereits in der Zielsetzung (1.1) erwähnt, wird besonders der Macro Average F_1-Score Wert betrachtet, da er einen guten ersten Eindruck bietet und Recall und Precision vereint. Für einen genaueren Einblick wird vor allem die Confusion Matrix herangezogen, um aufzuzeigen, bei welchen Klassen besonders viele falsche Klassifikationen vorkommen.

2.2 Zeitreihendaten

Da es sich bei den im Anwendungsszenario betrachteten Daten um die Aufzeichnung eines Datenstroms handelt, bei dem die Daten mit einem Zeitstempel versehen sind, sind die Datensätze Zeitreihendaten (Time Series Data). In einem Zeitreihen-Datensatz bietet der Zeitstempel die Möglichkeit, den Datensatz zeitlich zu ordnen, um so eine Veränderung in den Daten im Laufe der Zeit zu betrachten. Zeitreihen werden im maschinellen Lernen häufig für Vorhersagen von Trends genutzt. Dabei wird ein Sample als Input und das darauffolgende Sample als gewünschter Output genutzt. In dem hier betrachteten Anwendungsfall werden, um diese zeitliche Struktur nutzen und Informationen aus den Zeitreihendaten extrahieren zu können, Daten-Vorverarbeitungstechniken angewandt.

Daher wird das Konzept des *Sliding Window* genutzt. Das Sliding Window ist eine Technik, bei der ein kleiner Abschnitt der Daten, das *Window*, betrachtet wird. Dieses Window wird schrittweise über die Daten verschoben und der nächste Abschnitt wird betrachtet. Dabei gibt die Größe des Fensters bzw. Window (*window size*) an, wie viele Daten betrachtet werden. Dieses Konzept kann beim Feature Engineering verwendet werden.

2.2.1 Feature Engineering in einer Zeitreihe

Feature Engineering ist der Prozess aus Rohdaten Feature zu erzeugen oder zusätzliche Feature aus den bereits vorhandenen abzuleiten und zu erstellen, die dann ebenfalls Teil

eines Sample werden. Dabei gibt es bei Datensätzen zwei verschiedene Arten von Fetures, inhärente und abgeleitete Features. Die inhärenten Features sind bereits in den Daten enthalten und brauchen keinerlei Verarbeitung. Abgeleitete Feature sind Feature, die aus den bereits vorhandenen Daten durch Berechnung erzeugt werden können.
Gängige Features, die bei Zeitreihen abgeleitet werden, sind Zeit Feature, statische Feature und verzögerte Feature. Im Anwendungsszenario werden nur verzögerte Features (Lag- und Lead-Features) genutzt.

Verzögerte Feature Verzögerte Feature nutzen Feature aus vorherigen oder zukünftigen Samplen. Dabei wird unterschieden zwischen Lead- und Lag-Features. Ein Lead-Feature ist ein Feature, das aus einem späteren Datenpunkt zum aktuellen Sample hinzugefügt wurde. Es beschreibt das Einbeziehen von Wissen aus einem zukünftigen Zeitpunkt. Ein Lag-Feature ist ein Feature aus einem früheren Datenpunkt. Durch sie können Trends in den Daten erkannt werden. Lag-Features sind, anders als Lead-Features, in realen Anwendungsfällen zugänglich [Laz20].

2.2.2 Feature Engineering im Beispielszenario

Im Anwendungsszenario sind vor allem Lag-Features von Bedeutung. Beim Anwendungsszenario werden mithilfe des Sliding Window Konzeptes Lag-Features erstellt. Dabei wird zunächst festgelegt, wie groß die Schrittweite (step size) beim Erstellen der Features sein soll und daraufhin wird mit dem Wert der Fenstergröße (window size) bestimmt, wie viele Lag-Features erstellt werden soll.

Wenn step size=2 gesetzt wird, wird bei der Betrachtung jedes Mal ein Sample übersprungen. Die Window size gibt an, wie viele Lag-Feature erstellt werden sollen und daher wie viele Schritte gemacht werden.

#	mmsi	t	speed	speed_lag2	speed_lag4
1	124563552	1443650402	11.1	NaN	NaN
2	124563552	1443650432	10.2	NaN	NaN
3	124563552	1443650472	12.3	11.1	NaN
4	124563552	1443650415	7.4	10.2	NaN
5	124563552	1443650420	9.5	12.3	11.1

Tabelle 2.8: Window size und step size beides 2

Bei dem Datensatz in 2.8 wurden Lag Features für das Feature speed erstellt. Die window und step size wurde auf 2 gesetzt. Für das fünfte Sample wurden zwei neue Feature erstellt. Das Feature speed_lag2 ist der speed Wert des dritten Sample und das Feature

speed_lag4 hat den speed Wert des ersten Samples. Beim Erstellen der Werte wurden immer zwei Schritte gemacht (step size=2) und es wurden 2 Lag Feature erzeugt (window size=2). Gibt es nicht genügende Sample, können diese Feature jedoch nicht erzeugt werden. Durch diese Lag-Feature kann der zeitliche Verlauf eines oder mehrerer Feature betrachtet werden. Bei der Anwendung muss darauf geachtet werden, dass der Datensatz vor der Erzeugung dieser Feature zeitlich angeordnet ist und dass die Erzeugung für jedes Schiff einzeln passiert.

Ein konkreter Anwendungsfall soll im folgenden betrachtet werden. Betrachtet man den Datensatz 2.9, fällt auf, dass das Label des vierten Samples above ist, obwohl die Geschwindigkeit die gleiche wie im darauffolgenden Sample ist, welches mit normal gelabelt wurde.

#	mmsi	t	speed	movingSpeed
1	124563552	1443650402	11.0	above
2	124563552	1443650432	10.0	above
3	124563552	1443650472	12.0	above
4	124563552	1443650415	6.8	above
5	124563552	1443650420	6.8	normal
6	124563552	1443650430	6.8	normal
7	124563552	1443650437	8.7	normal

Tabelle 2.9: Beispielhafter Zeitreihendatensatz

Hier kann angenommen werden, dass eine Geschwindigkeit über mehr als nur ein Signal gehalten werden muss, um als Geschwindigkeitswechsel registriert zu werden.

#	mmsi	t	speed	speed_lag_1	label
1	124563552	1443650402	11.0	NaN	above
2	124563552	1443650432	10.0	11.0	above
3	124563552	1443650472	12.0	10.0	above
4	124563552	1443650415	6.8	12.0	above
5	124563552	1443650420	6.8	6.8	below
6	124563552	1443650430	6.8	6.8	below
7	124563552	1443650437	8.7	6.8	below

Tabelle 2.10: Beispielhafter Zeitreihendatensatz mit erzeugten Lag-Features

In 2.10 wurden für alle Sample daher Lag-Features erzeugt. Da nun der zeitliche Verlauf des speed Features in einem Sample vereint wurde, kann die Anforderung überprüft werden, dass die Geschwindigkeit über mehr als nur ein Signal gehalten werden muss [Laz20].

3 Maritimes Anwendungsszenario

Im maritimen Szenario sollen mittels Supervised Klassifikation Aktivitäten von Schiffen auf Basis von versendeten Signalen gelernt werden. Um das Lernen zu ermöglichen, müssen zunächst die Daten betrachtet werden, um sie daraufhin geeignet verarbeiten zu können.
Gemäß der Phase des *Data Understanding* sollen daher im Folgenden die zur Verfügung stehenden Daten, also die Signale und alle weiteren verfügbaren Daten, beschrieben werden, um ein Verständnis für die Datenmenge zu erlangen.

3.1 Datenerhebung

Bei den zur Verfügung stehenden Daten handelt es sich um AIS-Signale [RDCJ18][1]. Das Automatic Identification System (AIS) ist ein Maritime Reporting System d.h. ein Kommunikationssystem, mit dem Schiffe Informationen über ihren aktuellen Zustand mit anderen Schiffen und Receivern an Land innerhalb eines gewissen Sendebereichs per Funk teilen.
AIS-Signale werden automatisch in festgelegten Intervallen übertragen und beinhalten eine Reihe von statischen und dynamischen Informationen. Die statischen Informationen beinhalten unter anderem den Namen des Schiffs oder den Schiffstypen. Die dynamischen Informationen hingegen geben einen Einblick in den Zustand des Schiffs zum Zeitpunkt der Sendung. Die Geschwindigkeit oder die Position zählen unter anderem zu den dynamischen Signalen. Das AIS wurde ursprünglich zur Kollisionsvermeidung entwickelt, damit alle Schiffe innerhalb des Sendebereichs auf gefährliche Situationen reagieren können. Mittlerweile werden die Signale auch von Receivern an Land genutzt, um mithilfe der übermittelten Signale den Schiffsverkehr zu bewachen [AZ21].

Die Aufzeichnung der AIS-Signale fand in einem Zeitraum von sechs Monaten, vom 1. Oktober 2015 bis zum 31. März 2016 statt. Für die Erfassung der Signale wurde der Receiver des Naval Academy Research Institute in der französischen Hafenstadt Brest genutzt. Seine Position und der von ihm abgedeckte Bereich sind in Abbildung 3.1 eingezeichnet. Die Position des Receivers ist mit einem Stern markiert und sein abgedeckter

[1] https://zenodo.org/record/1167595

Bereich ist als blaues Polygon eingezeichnet. Der abgedeckte Bereich umfasst Teile der Keltischen See, des Nordatlantiks, des Ärmelkanals und des Golfs von Biskaya [RDC+19].

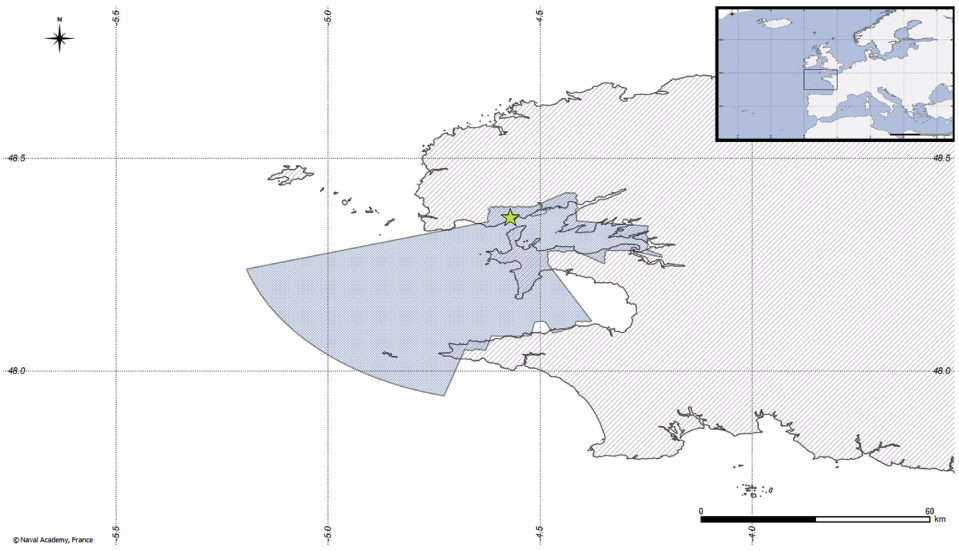

Abbildung 3.1: Position und Reichweite des Receivers in Brest aus [RDC+19]

Innerhalb des vom Reciever abgedeckten Bereichs konnten in den sechs Monaten über 19 Mio. Nachrichten von 5000 verschiedenen Schiffen gesammelt werden, von denen 1 Mio. statische AIS-Signale und 18 Mio. dynamische AIS-Signale sind [RDCJ18].

Um Aktivitätsmuster erkennen zu können, die von mehr als nur den Informationen aus den AIS-Signalen abhängen, wurden zusätzlich zu den AIS-Signalen weitere Daten aus verschiedenen Quellen einbezogen.
Die referenzierten Daten können in vier Kategorien aufgeteilt werden:

1. **Navigationsdaten:** Die Navigationsdaten enthalten die eben genannten statischen und dynamischen AIS-Signale.
2. **Schiffsbezogene Daten:** Die schiffsbezogenen Daten beinhalten genaue Informationen über alle registrierten Schiffe.
3. **Geografische Daten:** Geografische Daten ergänzen den Datensatz mit Koordinaten verschiedener Häfen sowie Regionen auf dem Meer wie z.B. Küstenregionen.
4. **Umweltrelevante Daten:** Die Umweltdaten beinhalten Informationen über das Wetter, die Windstärke und den aktuellen Zustand des Meeres.

Mit den 19 Mio. AIS-Signalen als Basis wurden diese Kategorien für alle weiteren Verarbeitungen genutzt [RDC+19].

3.2 Ereignisströme

Durch die Weiterverarbeitung der vier Datenkategorien wurden zwei Ereignisströme generiert, der *Spatial-Stream* (räumliche Ereignisstrom) und der *Critical-Stream*[2] (Bewegungsereignisstrom) [PAA+16]. Ereignisströme sind Datenströme, deren Datenpunkte Ereignisse darstellen. Mit Ereignis ist hier eine Zustandsveränderung zu einem Zeitpunkt gemeint [BD15]. Der Critical- und der Spatial-Stream beinhalten beide punktuelle Ereignisse, die Veränderungen im Verhalten, Zustand oder in der Position des Schiffs markieren.

Der Spatial-Stream beinhaltet Veränderungen in den räumlichen Beziehungen von Schiffen zueinander und zwischen Schiffen und gekennzeichneten Regionen. Da in den AIS-Signalen die Position eines Schiffes in Form von Koordinatenangaben gegeben ist und in den geografischen Daten Regionsgrenzen in Form von Polygonen enthalten sind, wurden Ereignisse erkannt, die eine Veränderung des Aufenthaltsortes eines Schiffs markieren. Zu den Ereignissen gehört das Betreten und Verlassen verschiedener Regionen sowie die Nähe zu anderen Schiffen. In den AIS-Signalen wurden ungefähr 374000 Signale als solche Ereignisse erkannt, die den Inhalt des Spatial-Streams ausmachen [PAD+19]. Der Spatial-Stream ist nicht öffentlich, daher ist es nicht möglich, Aktivitäten, die die geografische Position eines Schiffes betreffen, zu lernen.

Der zweite Ereignisstrom ist der Critical-Stream. Alle Ereignisse und daher alle Datenpunkte im Critical-Stream werden als *Critical Point* bezeichnet. Ein Critical Point bezeichnet einen spezifischen Zeitpunkt, der eine signifikante Änderung in der Bewegungsbahn eines Schiffes markiert. Dazu gehören Zeitpunkte, in denen ein Schiff seine Geschwindigkeit ändert oder aufhört, seine Position zu übermitteln. Es wurden in den AIS-Signalen neun verschiedene Ereignisse erkannt, die als Critical Point bezeichnet werden, welche im Abschnitt *3.2.1 Critical-Stream* näher erläutert werden sollen.

Mithilfe der beiden punktuellen Ereignisströme wurde ein durativer Ereignisstrom erzeugt, der Composite-Stream[3]. Dieser beinhaltet durative Aktivitäten, denen ein Schiff nachgehen kann [PAD+19]. Der Composite-Stream soll im Abschnitt *3.2.2 Composite-Stream* näher erläutert werden.

Die beiden Ereignisströme, der Critical-Stream und der Composite-Stream sind beide frei zugänglich, sodass ihre Inhalte betrachtet und genutzt werden können.

[2] https://zenodo.org/record/2563256
[3] https://zenodo.org/record/2557290

3.2.1 Critical-Stream

Der Critical-Stream ist eine komprimierte Abstraktion der AIS-Signale. Die Komprimierung kommt dadurch zustande, dass alle für die Bewegungsbahn der Schiffe nicht relevanten AIS-Signale entfernt wurden. In den AIS-Signalen wurden 4,6 Mio. Signale als Critical Point erkannt, was einer Kompressionsrate von 75% entspricht. Trotz der hohen Kompressionsrate kann anhand des Critical-Streams die Bewegung eines Schiffes verfolgt werden.

#	**Critical Point**	**Beschreibung**
1	gap_start	Das Schiff hat aufgehört, Signale zu senden. Das Ereignis markiert den Beginn einer Funkstille.
2	gap_end	Das Schiff hat nach einer Funkstille wieder angefangen, Signale zu senden.
3	slow_motion_start	Das Schiff beginnt damit, sich in einer langsamen Geschwindigkeit zu bewegen. Als langsam gilt eine Geschwindigkeit zwischen 0,5 und 5 Knoten.
4	slow_motion_end	Das Schiff hört auf, sich in einer langsamen Geschwindigkeit zu bewegen.
5	stop_start	Das Schiff beginnt damit stillzustehen. Die Geschwindigkeit wurde auf unter 0,5 Knoten reduziert.
6	stop_end	Das Schiff hört auf stillzustehen.
7	change_in_speed_start	Das Schiff startet damit, seine Geschwindigkeit zu ändern.
8	change_in_speed_end	Das Schiff hört auf, seine Geschwindigkeit zu ändern.
9	change_in_heading	Das Schiff ändert seine Richtung.

Tabelle 3.1: Übersicht über die Critical Points

Die Tabelle 3.1 soll einen Überblick darüber bieten, welche Critical Points bestimmt wurden und was sie aussagen [PAD+19]. Eine Besonderheit bei der Erkennung eines Critical Points ist es, dass zu jedem Zeitpunkt mehrere gleichzeitig stattfinden können. Ein Schiff kann sowohl damit beginnen anzuhalten (*stop_start*) als auch seine Geschwindigkeit zu ändern (*change_in_speed_start*). Um diese Signale markieren zu können, die mehrere Critical Points beinhalten, wird eine achtstellige Binärzahl verwendet. In der Tabelle 3.2 ist angegeben, welche Stelle der Binärzahl zu welchem Critical Point korrespondiert.

Binärzahl	Critical Point
00000001	stop_start
00000010	stop_end
00000100	slow_motion_start
00001000	slow_motion_end
00010000	gap_end
00100000	change_in_heading
01000000	change_in_speed_start
10000000	change_in_speed_end

Tabelle 3.2: Binärzahlen der Critical Points

Ist die Binärzahl beispielsweise 00100100 bedeutet es, dass ein Schiff seine Fahrtrichtung geändert hat (*change_in_heading*), da das sechste Bit gesetzt ist. Außerdem hat es sich im Vergleich zu seinem vorherigen Kurs verlangsamt (*slow_motion_start*), da das dritte Bit gesetzt ist. Der folgende Auszug aus dem Critical-Stream (3.3 und 3.4) enthält nur die Critical Points eines Schiffes [PK18].

#	id	timestamp	longitude	latitude	...
1	227142200	1443652394000	-4.5001965	47.82828	...
2	227142200	1443652398000	-4.500025	47.82834	...
...	...	...	...	...	...
36778	227142200	1454615075000	-4.324072	48.133556	...
...	...	...	...	...	...
73552	227142200	1459459077000	-4.3239217	48.09794	...
73553	227142200	1459459097000	-4.323913	48.0979	...

Tabelle 3.3: Auszug aus dem Critical-Stream (Teil 1)[PK18]

Im ersten Teil des Auszugs aus dem Critical Stream (3.3) sind folgende Informationen zu sehen:

id Die Identifikationsnummer, die im Critical Stream enthalten ist, ist die Maritime Mobile Service Identify Number (MMSI). In den AIS-Signalen ist die Nummer als *sourcemmsi* gekennzeichnet. Sie ist eine eindeutige Identifikationsnummer für Schiffe. Durch sie kann im Datensatz der zeitliche Verlauf der Bewegungen eines Schiffes betrachtet werden.

t Der Wert aus t stellt einen Zeitstempel dar, der den Zeitpunkt, zu dem das Signal gesendet wurde, in Unixzeit in Millisekunden angibt. Der Wert gibt die Anzahl der Millisekunden, die seit 00:00:00 UTC am 1. Januar 1970 vergangen sind, an.

longitude und latitude Die zwei Werte in *longitude* und *latitude* geben die Position des Schiffes in Längen- und Breitengraden an.

#	...	annotation	speed	heading	turn	course
1	...	10000	0	0	0	59.1
2	...	100000	7.015706946	0	127	60.8
...	...	...	...	...	...	...
36777	...	10000001	0.421945412	84.455885	-126	102.7
...	...	...	...	...	...	...
73552	...	1000010	0.641575336	220.9321675	0	348.9
73553	...	10000001	0.436828224	220.9321675	0	128.8

Tabelle 3.4: Auszug aus dem Critical-Stream (Teil 2)[PK18]

annotation Der Wert beschreibt die Ereignisse in Form einer Binärzahl, in dem die Critical Points markiert werden. Führende Nullen werden dabei entfernt.

speed Die Geschwindigkeit des Schiffes in Bezug auf den Boden. Diese entspricht nicht der Geschwindigkeit durch das Wasser, da Strömungen die tatsächliche Bewegung des Schiffes relativ zum Meeresboden beeinflussen können.

heading Die Richtung, in die ein Schiff relativ zum magnetischen oder geografischen Nordpol ausgerichtet ist.

turn Der Wert beschreibt den Winkel, den das Schiff pro Minute nach rechts oder links dreht.

course Die Bewegungsrichtung in Bezug auf den Boden, die ein Schiff relativ zum magnetischen Nordpol oder geografischen Nordpol genommen hat in Grad.

In dem Auszug aus dem Critical-Stream (3.3 und 3.4) wird deutlich, dass es sich bei ihm um angereicherte AIS-Signale handelt, da sie die Informationen aus den dynamischen AIS-Signalen enthalten [AZ21].

In den sechs Monaten der Datenerhebung konnten alleine bei einem Schiff 73553 Critical Points erkannt werden. Obwohl viele der AIS-Signale des Schiffs im Critical-Stream entfernt wurden, liegen die einzelnen Signale nur wenige Sekunden oder Minuten auseinander. In der Zeile 36777 ist zu sehen, dass laut dem Wert in *annotation* zu dem Zeitpunkt sowohl das Ende einer Veränderung in der Geschwindigkeit stattfand (*change_in_speed_end*) als auch das Schiff angefangen hat, stillzustehen (*stop_start*). Das Vorkommen dieser Signale, die mehrere Ereignisse darstellen, bilden im Critical-Stream mit 1,5 Mio ungefähr 34% der Ereignisse. Bis zu vier Critical Points konnten zu einem Zeitpunkt erkannt werden. Daher gibt es deutliche Überschneidungen zwischen den einzelnen Ereignissen [PK18].

3.2.2 Composite-Stream

Mit dem Spatial- und Critical-Stream als Grundlage konnte ein dritter Ereignisstrom erzeugt werden. Dieser Ereignisstrom wird als Composite-Stream bezeichnet.
Der Composite-Stream enthält durative Aktivitäten, denen ein Schiff nachgehen kann. Dazu beschreiben Pitsikalis et al. in *Composite Event Recognition for Maritime Monitoring*, dass die zwei Ereignisströme zusammen mit von Domänenexperten definierten Aktivitätsmuster als Basis genutzt wurden. Im Auszug 3.5 aus dem Composite-Stream ist der Aufbau einer solchen durativen Aktivität zu erkennen.

#	FluentName	MMSI	Argument	Value	T_start	T_end
1	withinArea	245257000	nearCoast	true	1443650403	1443694175
2	withinArea	245257000	nearPorts	true	1443650403	1443695038
...	...	...	...	...	...	...
1985185	changingSpeed	228051000	NaN	true	1452626209	1452626220
...	...	...	...	...	...	...
3970367	trawling	228037600	NaN	true	1459461578	1459461601
3970368	trawlSpeed	228037600	NaN	true	1459461578	1459461601

Tabelle 3.5: Auszug aus dem Composite-Stream[PA19]

Der Auszug enthält alle Informationen, die im Critical-Stream enthalten sind. Diese sollen im Folgenden näher erläutert werden.

T_start und T_end Die Zeitstempel *T_start* und *T_end* markieren den Start- und Endpunkt einer Aktivität. Anders als bei punktuellen Ereignissen werden die durativen Aktivitäten mit einem Zeitpunkt, an dem das Schiff diese Aktivität startet (*T_start*) und mit einem Zeitpunkt, an dem das Schiff die Aktivität beendet (*T_end*), definiert. Anders als im Critical-Stream sind die Zeitstempel in Unixzeit in Millisekunden angegeben.

FluentName Die Werte in *FluentName* bezeichnen eine der 21 verschiedenen Aktivitäten, die erkannt wurden.

Argument und Value Hier werden nähere Informationen zu den Aktivitätsmustern aus *FluentName* angegeben.

Im Datensatz sind 21 verschieden Werte für *FluentName*, die jeweils eine Aktivität beschreiben [PA19]. Die 21 erkannten Aktivitätsmuster, ihre Häufigkeit und ihre Beschreibung sind in der Tabelle 3.6 aufgeführt.

#	FluentName	Aufkommen	Beschreibung
1	movingSpeed	1293080	Bewegt sich gemessen an dem Standard für den jeweiligen Schiffstypen unterdurchschnittlich, durchschnittlich oder überdurchschnittlich schnell. Spezifiziert durch zusätzliche Werte in Value.
2	changingSpeed	777154	Verändert sein Tempo.
3	tuggingSpeed	524144	Hat die Geschwindigkeit eines Schiffs, das ein anderes Schiff zeiht. Die Geschwindigkeit liegt zwischen 1,2 und 15 Knoten.
4	stopped	379318	Steht still. Die Geschwindigkeit liegt zwischen 0 und 0,5 Knoten.
5	underWay	349864	Ist unterwegs.
6	lowSpeed	161069	Bewegt sich langsam fort d.h. mit einer Geschwindigkeit zwischen 0.5 und 5 Knoten.
7	withinArea	146114	Befindet sich in einer von fünf definierten Regionen.
8	trawlSpeed	98346	Hat die Geschwindigkeit eines Schleppernetzschiffes. Die Geschwindigkeit liegt zwischen 1 und 9 Knoten.
9	drifting	57281	Driftet vom Kurs ab
10	gap	55784	Zeitraum einer Funkstille
11	highSpeedNC	44622	Fährt zu schnell nahe der Küste.
12	sarSpeed	43010	Bewegt sich in der Geschwindigkeit eines Rettungsschiffes mit mindestens 2.7 Knoten.
13	trawlingMovement	14563	Bewegungsmuster ähnelt dem eines Schleppernetzschiffes.
14	anchoredOrMoored	12682	Ist verankert oder angebunden.
15	loitering	7125	Verweilt an einem Ort.
16	rendezVous	3548	Begegnet einem anderen Schiff.
17	sarMovement	1227	Bewegt sich wie ein Rettungsschiff.
18	pilotBoarding	730	Navigiert ein anderes Schiff.
19	trawling	432	Fischt mittels Schleppnetz.
20	sar	150	Führt eine Rettungsmission aus.
21	tugging	127	Schleppt ein anderes Schiff.

Tabelle 3.6: Die 21 Aktivitätsmuster des Composite-Streams [PA19][PAD+19]

Da es sich um echte Daten handelt, ist das Aufkommen der Aktivitäten im Composite-Stream repräsentativ zu der realen Situation auf See. Es ist zu erkennen, dass die

häufigsten Aktivitäten die Geschwindigkeit der Schiffe betreffen. Aktivitäten hingegeben, die nur von bestimmten Schiffstypen ausgeführt werden können, kommen seltener im Datensatz vor. Beispielsweise betreffen Aktivitäten wie *trawlSpeed*, *trawlingMovement* und *trawling* nur Schleppernetzschiffe.

Zudem hängen viele der Aktivitäten von mehreren Umständen, wie z.B. dem Schiffstypen und anderen Aktivitäten ab. Diese Abhängigkeiten sollen an der Aktivität *trawling* aufgezeigt werden. Die Aktivität ist wie folgt definiert:

1. In den statischen AIS-Signalen ist der Schiffstyp des Schiffes ein Trawler ($shiptype = trawler$).
2. Das Schiff befindet sich in einer Fischereiregion ($FluentName = withinArea$ mit $Argument = fishing$). Für diese Information wurden Ereignisse des Spatial-Streams genutzt.
3. Die Bewegungsbahn des Schiffes gleicht der Bewegung eines Schiffes beim Fischen mit einem Schleppernetz ($trawlingMovement$).
4. Die Geschwindigkeit des Schiffes ist die Geschwindigkeit, die ein Schiff beim Fischen hat ($trawlSpeed$).

Diese Umstände müssen für einen bestimmten Zeitraum gehalten werden, damit die Aktivität *trawling* erkannt werden kann [PAD+19]. Wie im Beispiel an *withinArea* zu erkennen ist, werden viele der Aktivitäten durch die Werte in $Value$ und $Argument$ konkretisiert. Die Werte, die sie annehmen können, sind in den Tabellen 3.7 und 3.8 aufgeführt.

Argument	**Vorkommen**
nearCoast	21802
nearPorts	36996
NaN	3819851
fishing	54508
anchorage	9548
natura	23260
<beliebige MMSI>	4405

Tabelle 3.7: Werte von *Argument* und ihre Anzahl im Composite-Stream [PA19]

Value	**Vorkommen**
true	2242188
nearPorts	282587
normal	478782
farFromPorts	152515
below	568955
above	245343

Tabelle 3.8: Werte von *Value* und ihre Anzahl im Composite-Stream [PA19]

Bei *Argument* handelt es sich um einen optionalen Wert. Hier sind Aussagen über die Position des Schiffes und seiner Umgebung enthalten. Der Wert konkretisiert demnach, dass das Schiff sich in dem von T_start und T_end eingeschlossen Zeitraum in einer

der definierten Regionen befindet. Wenn es sich bei dem Wert um eine MMSI handelt, bedeutet es, dass sich das Schiff in der Nähe des Schiffes befindet, welches durch diese MMSI gekennzeichnet wird.

Bei $Value$ handelt es sich um einen Wert, der die Aktivitätsmuster konkretisiert. Der Wert wird in jedem Fall, wo keine Konkretisierung nötig ist, auf *true* gesetzte. Ist eine Konkretisierung nötig, kann er die Werte aus der Tabelle 3.8 annehmen. Die Werte $nearports$ und $farFromPorts$ konkretisieren eine positionelle Information in Bezug auf die Nähe zu einem Hafen und die Werte *normal*, *below* und *above* werden bei der Aktivität *movingSpeed* genutzt, um zu beschreiben, ob die Geschwindigkeit für das Schiff als schnell, normal oder langsam gilt.

Bei vielen der Aktivitäten ist es auch eindeutig, dass sie mithilfe von nur zwei Critical Points erkannt wurden. Beispielsweise schließt die Aktivität *stopped* den Zeitraum zwischen den zwei Critical Points *stop_start* und *stop_end* ein. Das Gleiche gilt für die Aktivitäten *changingSpeed* (*change_in_speed_start* und *change_in_speed_end*) und *lowSpeed* (*slow_motion_start* und *slow_motion_end*) [PAD+19].

4 Datenvorbereitung

Gemäß der *Data Preparation* Phase des CRISP-DM Modells sollen die vorgestellten Daten so vorbereitet werden, dass ein Machine Learning Algorithmus ein Modell aus ihnen lernen kann. Hierzu gehört die Auswahl und Bereinigung der Daten, konstruktive Datenvorbereitung durch das Ergänzen oder Anpassen von Daten, Daten Zusammenführung und die Formatierung nach den Ansprüchen der Machine Learning Algorithmen. Da der Erfolg des Lernens von der Vorverarbeitung abhängt, muss der Datenvorbereitung besondere Aufmerksamkeit geschenkt werden.

Für die Vorbereitung der Daten wird die Python-Library Pandas[1] verwendet, die speziell für die effiziente Manipulation und Analyse tabellarischer Daten entwickelt wurde. Die wichtigste Datenstruktur in Pandas ist der *DataFrame*, der Datensätze in Tabellenform darstellt [WM10].

Für einige der Vorverarbeitungen und das spätere Lernen der Modelle, welches in Kapitel *5 Lernen der Modelle* beschrieben ist, wird die Open Source Library Scikit-learn[2] genutzt. Die in Scikit-learn implementierten Classifier setzen voraus, dass ein Datensatz gewisse Voraussetzungen erfüllt. Diese sollen wie folgt erfüllt werden:

1. Nachdem identifiziert wurde, welche Datensätze von Bedeutung sind, müssen diese in einem Datensatz zusammengefasst werden.
2. Im Datensatz darf es keine fehlenden Werte geben. Diese müssen daher entfernt oder verändert werden.
3. Da die Modelle mittels Supervised Learning Algorithmen gelernt werden sollen, müssen die Sample des Datensatzes gelabelt sein.
4. Der Datensatz sollte keine irrelevanten Features enthalten, da eine große Menge an Features zu einer hohen Modellkomplexität und Overfitting führen kann.
5. Der Datensatz sollte genügend Sample aus allen Klassen, vor allem aus der zu lernenden Klasse enthalten.

Sind all diese Voraussetzungen erfüllt, kann ein Modell trainiert werden [PVG+11]. Im Folgenden sollen daher alle Schritte beschrieben werden, die nötig sind, um die genannten Voraussetzungen zu erfüllen.

[1] https://pandas.pydata.org/
[2] https://scikit-learn.org/stable/

4.1 Zusammenführen der Datensätze

Um alle relevanten Feature, die nötig für das Erlernen eines Modells sind, einem Algorithmus übergeben zu können, müssen diese laut den Voraussetzungen in einem Datensatz zusammengefasst werden. Beim Zusammenfassen zweier Datensätze muss eine eindeutige Zuordnung eines Datenpunktes aus einem Datensatz zu einem anderen Datenpunkt aus einem zweiten Datensatz möglich sein. Die Zusammenführung findet daher immer anhand der MMSI und des Zeitstempels statt. Dadurch ist gewährleistet, dass es sich bei zwei Signalen nicht nur um zwei Signale des gleichen Schiffes handelt, sondern um das gleiche Signal.

Je nachdem, ob ein Ereignis des Critical Streams oder eine Aktivität des Composite Streams gelernt werden soll, unterscheidet sich die Vorgehensweise.
Da es sich bei dem Critical Stream um eine Abstraktion der AIS-Signale handelt und er die relevanten Informationen aus den AIS-Signalen enthält, muss keine weitere Information aus einem anderen Datensatz einbezogen werden, wenn ein Modell gelernt werden soll, dass einen Critical Point erkennt.

Wenn Aktivitäten des Composite-Streams gelernt werden sollen, wird der Critical-Stream mit dem Composite-Stream zusammengeführt. Ist der Schiffstyp relevant für die Aktivitäten, wird auch dieser aus den statischen AIS-Signalen hinzugefügt. Bei der Zusammenführung der beiden Datensätze werden alle Signale im Critical-Stream, deren Zeitstempel zwischen den Zeitstempeln aus dem Composite-Stream *T_start* und *T_end* liegt und deren MMSI übereinstimmt, zusammengeführt. Das Ergebnis der Zusammenführung ist auch das Labeln der Sample mit der jeweiligen Aktivität, der das Schiff zu dem Zeitpunkt der Signalsendung nachgeht.

4.2 Labeln der Sample

Da beim Supervised Learning der Algorithmus darauf angewiesen ist, dass die Daten gelabelt sind, müssen alle Sample gelabelt werden.

Beim Lernen von Ereignissen des Critical-Streams ist das Labeln bereits durch *annotation* erfüllt, jedoch besteht das Problem, dass zu einem Zeitpunkt mehrere Critical Points gleichzeitig stattfinden können. Wenn die Binärzahl in *annotation* beispielsweise 00101000 lautet, stellt das sowohl einen Richtungswechsel (*change_in_heading*) als auch den Zeitpunkt dar, in dem ein Schiff aufhört, langsam zu fahren (*slow_motion_end*). Um den Datensatz so anzupassen, dass die Label deutlich definiert sind, bieten sich drei Möglichkeiten an.

Die erste Möglichkeit besteht darin, dass alle Signale, die mehr als ein gesetztes Bit in *annotation* haben, entfernt werden. Dies würde jedoch zu einem Verlust von 34% der

Sample, aber auch zu einer klaren Trennung der Klassen führen. Die zweite Möglichkeit besteht darin, die Zeilen zu duplizieren und für jedes gesetzte Bit ein einzelnes Sample zu erstellen.

mmsi	t	annotation
468997711	1443650402	10000
124563552	1443650432	1000001
368942311	1443650472	100

Tabelle 4.1: Critical Point

mmsi	t	annotation
468997711	1443650402	10000
124563552	1443650432	1000000
124563552	1443650432	1
368942311	1443650472	100

Tabelle 4.2: Aufgeteilter Critical Point

Wird das Prinzip beispielsweise auf den Datensatz in 4.1 angewendet, ist das Ergebnis der Datensatz in der Tabelle 4.2. Bei der Duplizierung der Zeilen wird jedoch deutlich, dass dieses Vorgehen keine Vorteile bietet. Da jeder Classifier das Ziel hat, eine Verbindung zwischen den Features und der Klasse zu finden, ist ein solcher Datenpunkt mit den gleichen Featurewerten, aber unterschiedlichen Label problematisch.

Der dritte Ansatz ist es, beim Lernen die One vs. Rest Strategie zu nutzen. Soll beispielsweise ein Modell für das Ereignis *slow_motion_end* (*1000*) gelernt werden, werden alle Sample, die im Wert in *annotation* das vierte Bit gesetzt haben, zu *slow_motion_end* geändert. Alle anderen Critical Points werden zu einer Klasse, der Rest Klasse, zusammengefasst. Der Ansatz bietet den Vorteil, dass wenn für ein einzelnen Critical Point ein Modell gelernt wird, auch Sample korrekt klassifiziert werden können, die mehrere Critical Points darstellen. Beim Lernen der Critical Points wird daher immer dieser Ansatz gewählt.

Bei der in Abschnitt 4.1 beschriebenen Zusammenführung der Datensätze werden ausschließlich die Sample des Composite-Streams von der zu lernenden Aktivität verwendet. Alle anderen Sample werden aus dem Datensatz entfernt. Wenn beispielsweise die Aktivität *stopped* gelernt werden soll, werden alle Zeilen aus dem Composite-Stream entfernt, für die $FluentName \neq stopped$ gilt. Die folgenden Ausschnitte (4.3 und 4.4) zeigen das Ergebnis einer solchen Zusammenführung für ein bestimmtes Schiff.

#	mmsi	t	lon	lat	annotation	...
1	228037700	1443657660000	-4.39	48.20	100000	...
2	228037700	1443657681000	-4.39	48.20	10000001	...
3	228037700	1443657761000	-4.39	48.20	100000	...
4	228037700	1443657770000	-4.39	48.20	1000010	...
5	228037700	1443657780000	-4.39	48.20	10000000	...

Tabelle 4.3: Critical- und Composite-Stream zusammengeführt (Teil 1)

#	...	speed	heading	turn	course	FluentName
1	...	0.37	135.35	-127.0	149.8	NaN
2	...	0.43	135.35	-127.0	196.5	stopped
3	...	0.42	135.35	-127.0	251.1	stopped
4	...	0.51	135.35	-127.0	257.5	stopped
5	...	0.48	135.35	-127.0	263.0	NaN

Tabelle 4.4: Critical- und Composite-Stream zusammengeführt (Teil 2)

In dem Datensatz (4.3 und 4.4) nimmt bei dem zweiten Sample das Feature *annotation* den Wert 10000001 an. Das bedeutet, dass zu dem Zeitpunkt erkannt wurde, dass ein Schiff damit beginnt stillzustehen (stop_start). Das vierte Sample markiert mit *annotation = 1000010* den Critical Point *stop_end*. Entsprechend dazu sind die Sample, die von den zwei Critical Points eingeschlossen wurden, alle mit *stopped* gelabelt. Diese Art, die Sample zu labeln, wird jeweils für die Aktivität durchgeführt, zu der ein Modell gelernt werden soll, da es große zeitliche Überschneidungen zwischen den Aktivitäten gibt, weil ein Schiff mehreren Aktivitäten gleichzeitig nachgehen kann.

4.3 Kürzen der Datenmenge

Da die Daten über sechs Monate aufgezeichnet wurden, ist die Datenmenge auch dementsprechend groß. Je größer die Datenmenge ist, desto mehr Zeit nimmt auch für das Labeln und Erlernen von Modellen sowie die Auswertung in Anspruch. Das Ziel ist es daher, die Datenmenge so zu kürzen, dass genügend Sample der zu lernenden Klasse verbleiben.
Die Testdurchläufe beim Lernen der Modelle hat ergeben, dass bereits 2% des Critical-Streams ausreicht, um ein Modell zu lernen und es zu testen. Um eine realistische Stichprobe des gesamten Datensatzes zu erhalten, werden daher die Daten aufsteigend nach ihrem Zeitstempel sortiert und es werden die ersten 2% der Sample behalten. Im Critical-Stream sind dadurch immer noch ungefähr 90000 Sample enthalten.

Da beim Lernen der Aktivitäten aus dem Composite-Stream die beiden Datensätze zusammengeführt werden, werden auch hier nur 2% des Critical-Streams verwendet.
Da durch diese Art der Datenkürzung die Verteilung der Klassen realitätsnah bleibt, muss bei der Evaluation darauf geachtet werden, dass hohe Accuracy Werte nicht nur daher entstanden sind, dass die Sample hauptsächlich als die Mehrheitsklasse klassifiziert wurden.

4.4 Feature Selection

Ist der gelabelte Datensatz vorhanden und es wurde entschieden, welches Ereignis oder welche Aktivität gelernt werden soll, sollten die Features, die irrelevant sind, aus dem Datensatz entfernt werden. Um diese Features auszuwählen, werden die Definitionen der jeweiligen Aktivität oder des jeweiligen Ereignisses der Domänenexperten herangezogen.

Als Beispiel wird die Definition eines Critical Point genutzt. Der Critical Point *stop_start* sagt aus, dass ein Schiff so weit seine Geschwindigkeit über eine gewisse Zeitspanne gesenkt hat, dass sie zwischen 0.5 und 0 Knoten liegt. Demnach ist die Geschwindigkeit das ausschlaggebende Feature.
Die gleiche Vorgehensweise gilt bei den Aktivitäten des Composite-Stream. Beispielsweise ist bei *trawlSpeed* sowohl Geschwindigkeit als auch der Schiffstyp von Bedeutung. In der Definition ist festgelegt, dass die Geschwindigkeit zwischen 1 und 9 Knoten liegen und es sich bei dem Schiff um ein Trawler handeln muss. Daher muss in diesem Fall der Schiffstyp aus den statischen AIS-Signalen in den Datensatz eingefügt werden.

Scikit-Learn bietet zudem Funktionen, um Feature Importance zu bestimmen. Die Feature Importance sagt aus, welche Features die mit dem größten Einfluss für ein Modell bei der Klassifizierung sind. Eine Möglichkeit besteht darin, bei Machine Learning Algorithmen, die auf Entscheidungsbäumen basieren, die wichtigsten Feature anzeigen zu lassen. Daher wird der Random Forest Classifier genutzt, um einen Einblick darin zu gewinnen, welche Features schlussendlich relevant waren. Diese Ausgabe gilt nicht für alle Classifier, bietet jedoch einen Einblick in die Wichtigkeit der Features für den Random Forest Classifier [PVG+11].

4.5 Feature Engineering

Bei einigen der zu lernenden Ereignisse ist es, wie in Abschnitt *2.2 Zeitreihendaten* beschrieben, sinnvoll Lag-Features zu erzeugen, um die Veränderungen eines oder mehrere Features eines Schiffes über einen Zeitraum zu betrachten. Dabei ist die Wahl der Features, der Fenstergröße und der Schrittgröße individuell. Ob und welche Lag-Features erzeugt wurden, wird daher bei jedem Modell angegeben. Lead-Features werden nicht erzeugt, um die Realitätsnähe zu bewahren. Bei vielen Modellen kann ein hoher Performancewert auch ohne Lag-Features erreicht werden, jedoch steigern sie die Performance in allen Testfällen, in denen sie verwendet wurden. Beim Erstellen muss darauf geachtet werden, dass der Datensatz zeitlich angeordnet ist und dass die Lag-Features für jedes Schiff getrennt erstellt werden.

Ein Problem beim Nutzen der Lag-Features ist, dass kein konsistenter Abstand zwischen einzelnen Signalen gewährleistet ist. Da sich die Sendefrequenz bei einer höheren

Geschwindigkeit steigert und bei einer niedrigeren fällt und je nach Schiffstyp unterschiedlich ist, kann keine universelle Senderate bestimmt werden. Zudem wurden beim Critical Stream viele Signale entfernt und Funkstille ist ebenfalls enthalten. Es kann daher nicht gesagt werden, ob zwei Signale immer im gleichen Abstand versendet wurden und dementsprechend kann auch nicht eindeutig bestimmt werden, dass die Lag-Features für einen festen Zeitraum erstellt wurden.

4.6 Fehlende Werte

Viele der Datensätze enthalten fehlende Werte, jedoch sind diese nicht mit den Algorithmen kompatibel. Eine Möglichkeit beim Umgang mit fehlenden Daten ist es, Sample mit fehlenden Werten aus dem Datensatz zu entfernen. Die Zeilen mit fehlenden Features werden entfernt, da eine willkürliche Kodierung, keine Vorteile bietet, wenn das Feature von Bedeutung ist. Zudem entstehen beim Erzeugen von Lag-Features häufig fehlende Werte, da es vorkommen kann, dass es kein früheres Signal gibt, auf das zugegriffen werden kann. Um den Datensatz mit den Algorithmen kompatibel zu machen, werden die Sample mit fehlenden Werten entfernt.

4.7 Feature Scaling

Einige Machine Learning Algorithmen arbeiten besser, wenn die Werte der Features skaliert werden. Daher werden für die Algorithmen, wo die Skalierung die Performance steigert, die Werte der Features skaliert. Bei der genutzten Skalierungsmethode werden die Werte in den Bereich [0, 1] oder [-1, 1] skaliert. Das Intervall [0, 1] wird gewählt, wenn im Feature nur positive Werte enthaltenden sind. Das Intervall [-1, 1] wird verwendet, wenn auch negative Werte enthalten sind. In den ausgewählten Algorithmen ist diese Skalierung nur für einen von ihnen relevant, dem Support Vector Classifier [PVG+11].

4.8 Aufteilen in Trainingsdaten und Testdaten

Wurden alle nötigen Vorverarbeitungsschritte durchgeführt, kann der gelabelte Datensatz anschließend in die Trainings- und Testdaten aufgeteilt werden. Dafür wird der gesamte Datensatz zunächst in alle Sample X und alle zugehörigen Label y aufgeteilt. Danach werden diese aufgeteilt in Trainingsdaten (X_{train}, y_{train}) und Testdaten (X_{test}, y_{test}). Um sicherzustellen, dass die Verteilung der Klassen im Trainings- und Testdatensatz repräsentativ für den Datensatz sind, wird Stratified Sampling genutzt. Dadurch

soll sichergestellt werden, dass alle Klassen angemessen in Test- und Trainingsdatensatz vertreten sind. Diese Art von Sampling ist besonders relevant, wenn die Größen der Klassen unausgeglichen sind. Sind 10% der Sample aus der positiven Klasse und 90% der Sample aus der negativen Klasse, kann es bei einer zufälligen Aufteilung dazu kommen, dass im Trainingsdatensatz nur wenige Sample der positiven Klasse enthalten sind und so das Modell nicht in der Lage ist, eine geeignete Klassifikationsfunktion zu finden. Das Stratified Sampling stellt sicher, dass das ursprüngliche Verhältnis der Klassengrößen im gesamten Datensatz, in den Testdaten und den Trainingsdaten erhalten bleibt [PVG+11]. Dadurch bleibt auch die Realitätsnähe des Datensatzes erhalten.

5 Lernen der Modelle

Nachdem die Datenvorbereitung abgeschlossen ist, folgt die *Modeling* Phase, in der Algorithmen gewählt und mit den Trainingsdaten Modelle gelernt werden. Daraufhin werden die Modelle, gemäß der Evaluation Phase, anhand von den in *2.1.6 Modell Evaluation* beschriebenen Metriken evaluiert.
Wie in Kapitel *4 Datenvorbereitung* beschrieben, wird zum Lernen der Modelle die Python Library Scikit-Learn genutzt. In der Bibliothek ist eine Vielzahl an Supervised Klassifikations Algorithmen implementiert. Einer der größten Vorteile bietet die konsistente API. Diese Konsistenz ermöglicht das einfache Verwenden und Vergleichen verschiedener Algorithmen. Zudem sind für alle Evaluationsmetriken Berechnungen implementiert. Einen weiteren Vorteil bietet die Möglichkeit, dass ein Modell mehrfach mit verschiedenen Parametern trainiert werden kann und die Parameter ausgegeben werden können, die die besten Ergebnisse erzielt haben. Trotz der Möglichkeit, in Scikit-Learn bei einer Mehrklassen-Klassifizierung mehrere binäre Klassifizierungen durchzuführen und die Ergebnisse zusammenzufassen, wird bei jedem Modell die One vs. Rest Methode genutzt. Das ermöglicht eine bessere Übersicht über die Leistungen der Modelle [PVG+11].

Das Lernen aller Modelle wurde auf der Online Plattform kaggle[1] mit 16 Prozessoren und 30 GB RAM durchgeführt.
Durch das Fehlen des Spatial-Streams können alle Aktivitäten, die den Aufenthaltsort und die Entfernung von einem Schiff zu einem anderen betreffen, nicht gelernt werden. Auch alle Ereignisse, die eine Funkstille beschreiben, konnten nicht gelernt werden.
Das beinhaltet:

- withinArea
- highSpeedNC
- anchoredOrMoored
- rendezVous
- trawling
- gap
- gap_start

[1] https://www.kaggle.com/

- rendezVous
- tugging
- sar
- pilotBoarding
- anchoredOrMoored
- loitering

5.1 Auswahl der Algorithmen und Hyperparameteroptimierung

Scikit-Learn kategorisiert die überwachten Klassifizierungsalgorithmen in siebzehn verschiedene Kategorien. Aus diesen Kategorien wurden fünf gewählt, aus denen wiederum jeweils ein Classifier gewählt wurde. Aus den Ensemble Algorithmen wurde Random Forest (RF) gewählt, aus den Support Vector Machines der Support Vector Classifier (SVC), aus den Modellen, die Nachbarn betrachten, wurde k-Nearest Neighbors Classifier (k-NN) gewählt und aus den Diskriminanzanalyse Verfahren Linear Discriminant Analysis (LDA). Für alle Tests, in denen der SVC Classifier genutzt wurde, wurden die Werte jedes Features mit der in *4.7 Feature Scaling* beschriebenen Methode skaliert, damit der Classifier bessere Ergebnisse durch die Skalierung erzielen kann.

Jeder der Algorithmen hat Parameter, die eingestellt werden können. Diese unterschieden sich je nach Classifier und habe unterschiedliche Effekte. Scikit-Learn bietet mit der GridSearchCV Funktion eine Möglichkeit, die besten Hyperparameter für einen Algorithmus zu finden. Dabei werden für jeden Parameter eines Classifiers eine Menge an Werten festgelegt und jede mögliche Kombination der Parameter wird genutzt, das Modell wird gelernt, evaluiert und die Parameterwerte, die die besten Ergebnisse erzielt haben, werden ausgewählt [PVG+11]. Diese Hyperparameteroptimierung wurde für jedes gelernte Modell durchgeführt und die besten Parameter wurden festgelegt.

Im Folgenden soll die Vorgehensweise beim Lernen der einzelnen Modelle kurz erläutert werden. Die Modelle sollen evaluiert und die Ergebnisse kurz erläutert werden. Bei jedem gelernten Modell wird kurz erwähnt, welche Vorverarbeitungsschritte durchgeführt wurden, wie die Verteilung der Klassen ist und die Modelle werden evaluiert. Wenn bei der Evaluation die Werte Precision, Recall und F_1-Score genannt werden, ist immer der Macro Average gemeint. Alle Werte werden auf Zweinachkommastellen gerundet.

5.2 Punktuelle Ereignisse des Critical-Streams

Die Ereignisse des Critical-Streams, die gelernt wurden, sind:

- stop_start
- stop_end
- slow_motion_start
- slow_motion_end

Da sich das Vorgehen bei allen Critical Points gleicht, wird beim ersten Critical Point *stop_start* ausführlicher auf das Vorgehen und die Ergebnisse eingegangen. Bei allen weiteren wird jedoch nur kurz auf die relevanten Aspekte eingegangen.

stop_start

Um ein Modell zu lernen, das das Ereignis stop_start erkennen kann, wird nur der Critical-Stream genutzt. Nachdem der Datensatz auf 2% gekürzt und die Sample so gelabelt wurden, dass aus der Mehrklassen-Klassifizierung eine binäre Klassifizierung entsteht, wird die Definition des Ereignisses betrachtet. Laut der Definition von stop_start markiert das Ereignis den Beginn der Aktivität stopped. Das Schiff beginnt also damit, seine Geschwindigkeit auf unter 0,5 Knoten zu reduzieren [PAD+19].
Nach der Betrachtung der Definition wurde die Auswahl der Features und Feature Engineering durchgeführt und beim Aufteilen der Daten in Trainings- und Testdaten wurde Stratified Sampling verwendet, um die Klassenverteilung bei der Aufteilung beizubehalten. Abschließend wurden vier verschiedene Tests durchgeführt, mit jeweils einer unterschiedlichen Auswahl an Features. Diese Tests wurden auch für alle anderen Ereignisse des Crtical Streams durchgeführt. Die ausgewählten Features waren in den Tests:

1. sourcemmsi, t, speed
2. speed
3. speed, speed_lag1, speed_lag2, speed_lag3
4. sourcemmsi, t, speed, speed_lag1, speed_lag2, speed_lag3

In allen Testfällen, bis auf den zweiten, wurde versucht zu beachten, dass stop_start aussagt, dass die vorherige Geschwindigkeit des Schiffs höher war als 0,5 Knoten. Im ersten Testfall wird getestet, ob es möglich ist, diese zeitliche Entwicklung der Geschwindigkeit über die MMSI und den Zeitstempel einzubeziehen und in den letzten zwei Testfällen wurde diese Entwicklung mithilfe von Lag-Features in jedem Sample zusammenzufassen. Wird zu dem Random Forest Classifier angezeigt, welche Features bei dem von ihm gelernten Modell den größten Einfluss auf die Klassifizierung hatten, werden speed und

speed_lag1 angezeigt. Das Erstellen der Lag-Features hat daher eindeutig einen positiven Effekt auf die Performance des Modells. Dieser Effekt lässt sich auch erkennen, wenn man die Ergebnisse des ersten und zweiten Tests vergleicht. Die Ergebnisse des ersten Tests sind in 5.1 aufgelistet.

	RF	SVC	k-NN	LDA
Accuracy	0,92	0,91	0,89	0,91
Precision	0,76	0,46	0,58	0,46
Recall	0,73	0,50	0,54	0,50
F_1-Score	0,74	0,48	0,55	0,48

Tabelle 5.1: Evaluation der Modelle (stop_start)

Wie auch im ersten Test konnte in allen Tests ein F_1-Score von mindestens 74% erreicht werden, jedoch wurden die besten Ergebnisse im dritten Test erzielt. In dem dritten Test wurden drei Lag-Features erstellt, mit einer Schrittweite von eins und der Zeitstempel sowie die MMSI wurden entfernt. Das Vorgehen im dritten Testfall hat zu der folgenden Aufteilung der Klassen geführt:

Klasse	**Im Datensatz**	**Im Trainingsdatensatz**	**Anteil**
Rest	84740	8274	91,4%
stop_start	8013	795	8,6%

Tabelle 5.2: Verteilung der Klassen (stop_start)

In der Tabelle 5.2 ist zu sehen, dass die Klassen stark unausgeglichen sind und nur 8,6% der Sample zu der Klasse stop_start gehören. Trotz dieser unausgeglichenen Verteilung konnten der gewünschte F_1-Score von 85% erreicht werden. Beim Testen der Modelle ergab diese Abfolge von Vorverarbeitungsschritten folgende Ergebnisse:

	RF	SVC	k-NN	LDA
Accuracy	0,99	0,95	0,98	0,91
Precision	0,99	0,86	0,95	0,46
Recall	0,98	0,83	0,92	0,50
F_1-Score	0,98	0,84	0,94	0,48

Tabelle 5.3: Evaluation der Modelle (stop_start, Lag-Features)

In der Tabelle 5.3 ist aufgeführt, dass der höchste F_1-Score mit den Algorithmen Random Forest und k-NN erzielt wurden. Im Vergleich zu den Ergebnissen des ersten Tests (5.1) konnten die Werte von drei der vier Modelle deutlich gesteigert werden.

	Klasse stop_start	Klasse Rest
Klassifiziert als stop_start	6950	213
Klassifiziert als Rest	161	74676

Tabelle 5.4: Confusion Matrix (stop_start, Random Forest)

	Klasse stop_start	Klasse Rest
Klassifiziert als stop_start	6413	750
Klassifiziert als Rest	438	74399

Tabelle 5.5: Confusion Matrix (stop_start, k-NN)

Die Confusion Matrix (5.4) des Tests mit dem vom Random Forest Classifiers gelernten Modell zeigt, dass von den Sample der Klasse stop_start nur 161 falsch klassifiziert wurden. In der Confusion Matrix des k-NN Models (5.5) wurden jedoch 438 Sample falsch klassifiziert. Beide Modelle sind eindeutig in der Lage, auch bei einem realitätsnahen Datensatz mit unausgeglichenen Klassengrößen Sample korrekt zu klassifizieren. Der Random Forest Classifier erzielt jedoch die besten Ergebnisse.

stop_end

Das Vorgehen bei dem Lernen eines Modells, dass das Ereignis stop_end erkennen kann, ist analog zu dem Vorgehen beim Lernen eines Modells für stop_start. Auch hier wurden alle Vorverarbeitungsschritte durchgeführt und die Definition des Ereignisses herangezogen, um verschiedene Tests durchzuführen. Da die Definition aussagt, dass stop_end den Zeitpunkt markiert, an dem ein Schiff aufhört stillzustehen, daher die Geschwindigkeit auf über 0,5 Knoten steigt, wurden auch hier in den vier Testfällen die gleichen Features erstellt und genutzt [PAD+19]. Auch hier konnte mit drei Lag-Features mit einer Schrittweite von eins und ohne dem Zeitstempel und die MMSI die besten Ergebnisse erzielt werden. Auch die Verteilung der Klassen sind nahezu gleich und daher, wie in 5.6 zu sehen ist, unausgeglichen.

Klasse	**Im Datensatz**	**Im Trainingsdatensatz**	**Anteil**
Rest	85271	16822	91,9%
stop_end	7482	1496	8,1%

Tabelle 5.6: Verteilung der Klassen (stop_end)

Beim Testen der Modelle ergaben sich ebenfalls ähnliche Werte bei dem Random Forest und bei dem k-NN Classifier.

	RF	SVC	k-NN	LDA
Accuracy	0,99	0,92	0,99	0,92
Precision	0,99	0,46	0,98	0,46
Recall	0,99	0,50	0,98	0,50
F_1-Score	0,99	0,48	0,98	0,48

Tabelle 5.7: Evaluation der Modelle (stop_end)

In der Tabelle 5.7 ist zu sehen, dass der höchste F_1-Score auch hier mit den Algorithmen Random Forest und k-NN erzielt wurde. Die Ergebnisse bei dem SVC Classifier haben sich jedoch deutlich verschlechtert. Diese Werte kommen dadurch zustande, dass jedes Sample des Testdatensatzes von dem SVC Classifier als die Mehrheitsklasse klassifiziert wurde.
Die beiden Modelle, die vom Random Forest Classifier und vom k-NN Classifier gelernt wurden, sind auch hier eindeutig in der Lage, Sample korrekt zu klassifizieren.

slow_motion_start

Auch bei den Modellen, die den Critical Point slow_motion_start erkennen sollen, wurde die eben beschriebene Vorgehensweise genutzt. Die Definition sagt aus, dass es den Zeitpunkt beschreibt, in dem ein Schiff seine Geschwindigkeit auf unter 5 Knoten reduziert hat, deshalb wurden auch hier die gleichen Tests durchgeführt [PAD+19]. Anders als bei stop_start und stop_end konnte in dem Test, in denen keine Lag-Features erstellt wurden und die MMSI und der Zeitstempel erhalten blieben, keine F_1-Score Werte höher als 55% erreicht werden (5.8).

	RF	SVC	k-NN	LDA
Accuracy	0,96	0,96	0,95	0,96
Precision	0,81	0,48	0,59	0,48
Recall	0,52	0,50	0,54	0,50
F_1-Score	0,52	0,49	0,55	0,49

Tabelle 5.8: Evaluation der Modelle (slow_motion_start)

In der Tabelle 5.8 wird deutlich, dass keines der Modelle basierend auf der Geschwindigkeit, der MMSI und dem Zeitstempel in der Lage war, die Sample im Testdatensatz korrekt zu klassifizieren. In der Confusion Matrix 5.9 des Tests mit dem Random Forest Classifier, der bei den vorherigen Critical Points am besten abgeschnitten hat, wird deutlich, dass nahezu immer ein Sample als die Mehrheitsklasse klassifiziert wurde.

	Klasse slow_motion_start	Klasse Rest
Klassifiziert als slow_motion_start	52	27
Klassifiziert als Rest	1507	35515

Tabelle 5.9: Confusion Matrix (slow_motion_start, Random Forest)

Durch das Erzeugen von Lag-Features kann die Performance des Random Forest Classifiers und der k-NN Classifiers gesteigert werden.

	RF	SVC	k-NN	LDA
Accuracy	0,99	0,96	0,98	0,96
Precision	0,98	0,48	0,93	0,55
Recall	0,98	0,50	0,87	0,50
F_1-Score	0,98	0,49	0,90	0,50

Tabelle 5.10: Evaluation der Modelle (slow_motion_start, Lag-Features)

In 5.10 ist zu sehen, dass durch die Lag-Features die Modelle ihren F_1-Score nahezu verdoppeln konnten. Auch die Confusion Matrix in 5.11 des Random Forest Classifiers zeigt, wie gut das Modell arbeitet.

	Klasse slow_motion_start	Klasse Rest
Klassifiziert als slow_motion_start	1698	58
Klassifiziert als Rest	56	39103

Tabelle 5.11: Confusion Matrix (slow_motion_start, Random Forest)

Trotz der stark unausgeglichenen Klassenverteilung kann das Modell den Sample des Testdatensatzes korrekt als Critical Point slow_motion_start klassifizieren.

slow_motion_end

Bei den zu dem Cirtical Point slow_motion_end gelernten Modellen konnten ähnliche Ergebnisse wie in den Modellen zu slow_motion_start erzielt werden. Auch hier war das Ergebnis des Tests ohne Lag-Features mit einem maximalen F_1-Score von 55% nicht ausreichend. Nachdem die Lag-Features erzeugt und die MMSI und der Zeitstempel entfernt wurden, konnten ähnliche Werte wie zuvor erreicht werden.

	RF	SVC	k-NN	LDA
Accuracy	0,99	0,96	0,98	0,96
Precision	0,96	0,48	0,91	0,59
Recall	0,93	0,50	0,79	0,50
F_1-Score	0,94	0,49	0,84	0,50

Tabelle 5.12: Evaluation der Modelle (slow_motion_end, Lag-Features)

Auch hier konnte der Random Forest Calssifier das beste Ergebnis erzielen, gefolgt vom k-NN Classifier.

Übersicht

Da alle vier Critical Points eine Geschwindigkeitsveränderung zu einem Zeitpunkt markieren, waren die Vorverarbeitungsschritte für alle gleich. Auch die erzeugten und ausgewählten Feature waren bei allen die gleichen mit speed, speed_lag1, speed_lag2 und speed_lag3. Auch die Verteilung der Klassen war in jedem der Tests sowohl im Trainings- als auch im Testdatensatz unausgeglichen und dadurch repräsentativ für den gesamten Datensatz. Die Ergebnisse waren für den Random Forest Classifier und dem k-NN Classifier am besten.

#	**Critical Point**	**Accuracy**	**Precision**	**Recall**	**F_1-Score**	**Classifier**
1	stop_start	0,99	0,99	0,99	0,98	Random Forest
2	stop_end	0,99	0,99	0,99	0,99	Random Forest
3	slow_motion_start	0,99	0,98	0,98	0,98	Random Forest
4	slow_motion_end	0,99	0,96	0,93	0,94	Random Forest

Tabelle 5.13: Ergebnisse der Modelle (Critical-Stream)

In 5.13 ist zu erkennen, dass jedes der Modelle des Random Forest Classifiers mit einer sehr hohen Genauigkeit in der Lage war Sample korrekt zu klassifizieren. Die Ergebnisse des K-NN Classifiers gleichen denen des Random Forest Classifiers, jedoch waren die Ergebnisse des Support Vector Classifiers und des Diskriminanzanalyse Verfahrens nicht in der Lage, den gewünschten F_1-Score zu erreichen.

5.3 Aktivitäten des Composite-Streams

Die Aktivitäten des Composite-Streams, für die Modelle trainiert werden sollen sind:

- stopped
- lowSpeed
- trawlSpeed
- sarSpeed

Hierbei wurde das Ziel von einem F_1-Score von mindestens 85% nicht in jedem Fall erreicht. Auch beim Lernen der Modelle zu den Aktivitäten des Composite-Streams wird bei der ersten Aktivität stopped das Vorgehen genauer beschrieben.

stopped

Anders als beim Lernen der Modelle, die Ereignisse des Critical-Stresms erkennen sollen, werden hier zunächst beide Datensätze zusammengeführt, wie im Abschnitt *4.1 Zusammenführen der Datensätze* beschrieben wurde. Durch diese Zusammenführung sind die genutzten Datensätze wie in *4.2 Labeln der Sample* gelabelt worden. Bei allen Modellen wurde der Datensatz auf 2% gekürzt und beim Aufteilen in Test und Trainingsdaten wurde Stratified Sampling verwendet. Wie auch bei den Critical Points wie in 5.14 zu sehen ist, sind die Klassen in dem Datensatz unausgeglichen.

Klasse	**Im Datensatz**	**Im Trainingsdatensatz**	**Anteil**
Rest	71873	7028	77,5%
stopped	20880	2083	22,5%

Tabelle 5.14: Verteilung der Klassen (stopped)

Nach der Aufteilung des Datensatzes in Trainings- und Testdaten wurden vier Tests durchgeführt mit jeweils unterschiedlichen Features. Die Features in den vier Tests waren die folgenden:

1. sourcemmsi, t, annotation, speed
2. sourcemmsi, t, speed
3. sourcemmsi, t, annotation, speed, speed_lag1, speed_lag2, speed_lag3
4. speed, speed_lag1, speed_lag2, speed_lag3

Bei der Auswahl der Features wurde darauf geachtet, dass die Aspekte in der Definition zu der Aktivität stopped einbezogen werden. Ein Schiff ist im Zustand stopped zwischen den zwei Cirtical Points stop_start und stop_end. Die Geschwindigkeit des Schiffs liegt daher über einen Zeitraum hinweg zwischen 0,5 und 5 Knoten [PAD+19]. Die Crticial Points in annotation, die Geschwindigkeit sowie die MMSI und der Zeitstempel sind daher relevant. In allen Tests konnte ein F_1-Score von mindestens 75% erzielt werden. Die besten Ergebnisse konnten im vierten Test erzielt werden.

	RF	SVC	k-NN	LDA
Accuracy	0,99	0,90	0,98	0,77
Precision	0,98	0,85	0,97	0,39
Recall	0,99	0,93	0,97	0,5
F_1-Score	0,99	0,88	0,97	0,44

Tabelle 5.15: Evaluation der Modelle (stopped, Lag-Features)

Die Modelle im vierten Test haben die in 5.15 aufgelisteten Werte erreicht. Anders als die Modelle zu den Critical Points konnte hier auch der Support Vector Classifier einen F_1-Score von 88% erreichen. Da in allen Tests ähnlich hohe Werte erreicht werden, konnten durch Feature Engineering die Werte nur um 5%-10% erhöht werden. Gemessen am F_1-Score konnten daher auch hier Modelle erfolgreich gelernt werden.

lowSpeed

Die Definition von lowSpeed, sagt aus, dass die Critical Points slow_motion_start und slow_motion_end den Beginn und das Ende der Aktivität markieren. Die Geschwindigkeit liegt daher zwischen 0,5 und 5 Knoten [PAD+19]. Da die beiden Aktivitäten sich nur in dem Geschwindigkeitsintervall unterscheiden, sind die Datenvorbereitung und die Tests in den Modellen für die Aktivität lowSpeed, genauso wie die Tests zu stopped aufgebaut. Auch hier hat der vierte Test die besten Ergebnisse erzielt.

	RF	SVC	k-NN	LDA
Accuracy	0,94	0,79	0,92	0,79
Precision	0,90	0,39	0,87	0,49
Recall	0,92	0,50	0,89	0,50
F_1-Score	0,91	0,44	0,88	0,49

Tabelle 5.16: Evaluation der Modelle (lowSpeed, Lag-Features)

Wie in der Tabelle 5.16 zu erkennen ist, gleichen sich auch die Ergebnisse. Ein Unterschied besteht jedoch darin, dass der F_1-Score das Modell des Support Vector Classifiers

sich im Vergleich zum Modell zu stopped halbiert hat. Die Modelle des Random Forest Classifiers und des k-NN Classifiers erfüllen jedoch weiterhin die Anforderung über einen F_1-Score von über 85%.

trawlSpeed

Die Definition von trawlSpeed sagt aus, dass die Geschwindigkeit des Schiffs zwischen 1 und 9 Knoten liegen und dass es sich bei dem Schiff um ein Fischereischiff handeln muss [PAD+19]. Anders als bei den Modellen zu stopped und lowSpeed muss bei der Aktivität trawlSpeed daher auch der Schiffstyp aus den statischen AIS-Signalen in den Datensatz hinzugefügt werden. Die Schiffstypen wurden zuvor in Kategorien eingeteilt. Diese Kategorien beinhalten unter anderem Such- und Rettungsschiffe, Segelschiff und Fischereischiffe. Außerdem werden der Beginn und das Ende beider Aktivitäten nicht durch zwei Critical Points markiert.
Die Featureauswahl wurde in den Tests daher wie folgt angepasst:

1. sourcemmsi, t, speed, shiptype
2. sourcemmsi, t, speed, speed_lag1, speed_lag2, speed_lag3, shiptype
3. speed, speed_lag1, speed_lag2, speed_lag3, shiptype

Die Ergebnisse des Tests mit den besten Ergebnissen ist in der Tabelle 5.17 aufgeführt.

	RF	SVC	k-NN	LDA
Accuracy	0,96	0,93	0,96	0,93
Precision	0,93	0,47	0,84	0,47
Recall	0,74	0,50	0,82	0,50
F_1-Score	0,80	0,48	0,83	0,48

Tabelle 5.17: Evaluation der Modelle (trawlSpeed, Lag-Features)

Der höchste F_1-Score wurde mit dem Modell des Random Forest Classifiers erreicht, jedoch wurde der gewünschte Wert von 85% nicht erreicht.

Die Gründe für diese Ergebnisse konnten nicht ermittelt werden. Im Trainingsdatensatz sind über 7000 Samplen mit dem Label trawlSpeed enthalten. Die Klasse ist daher ausreichend im Datensatz repräsentiert. Zudem hat das selbe Vorgehen bei der Aktivität sarSpeed zu besseren Ergebnissen geführt.

sarSpeed

Auch bei der Aktivität sarSpeed ist in der Definition die Geschwindigkeit und der Schiffstyp von Bedeutung. Die Definition sagt aus, dass die Geschwindigkeit des Schiffs bei mindestens 2,7 Knoten liegen muss und dass es sich um ein Such- oder Rettungsschiff handeln muss [PAD+19].
Anders als in den Modellen zu trawlSpeed konnte sowohl mit als auch ohne Lag-Features ein F_1-Score von über 85% erreicht werden.

	RF	SVC	k-NN	LDA
Accuracy	0,99	0,90	0,97	0,90
Precision	0,99	0,45	0,85	0,74
Recall	0,96	0,50	0,85	0,57
F_1-Score	0,97	0,47	0,85	0,63

Tabelle 5.18: Evaluation der Modelle (sarSpeed)

Die Tabelle 5.18 zeigt die Ergebnisse, die erreicht wurden konnten, wenn die Features sourcemmsi, t, speed und shiptype im Datensatz enthalten sind.

	RF	SVC	k-NN	LDA
Accuracy	0,96	0,89	0,98	0,89
Precision	0,89	0,44	0,95	0,44
Recall	0,91	0,5	0,97	0,5
F_1-Score	0,90	0,47	0,96	0,47

Tabelle 5.19: Evaluation der Modelle (sarSpeed, Lag-Features)

Die Ergebnisse in der Tabelle 5.19 wurden mit den Features speed, speed_lag1, speed_lag2, speed_lag3 und shiptype erzielt. In beiden Tests konnten die beiden Classifier, Random Forest und k-NN, den gewünschten F_1-Score erreichen.

Übersicht

#	**Aktivität**	**Accuracy**	**Precision**	**Recall**	**F_1-Score**	**Classifier**
1	stopped	0,99	0,98	0,99	0,99	Random Forest
2	lowSpeed	0,94	0,90	0,92	0,91	Random Forest
3	trawlSpeed	0,96	0,84	0,82	0,83	k-NN
4	sarSpeed	0,99	0,99	0,96	0,97	Random Forest

Tabelle 5.20: Ergebnisse der Modelle (Composite-Stream)

In der Tabelle 5.20 ist zu erkennen, dass der Random Forest Classifier in drei der vier Fälle die besten Ergebnisse erzielen konnte und die Modelle mit einer hohen Genauigkeit in der Lage sind, Sample korrekt zu klassifizieren. In den drei Fällen war der k-NN Classifier ebenfalls in der Lage, Modelle zu lernen, die einen F_1-Score von über 85% erreicht haben. Bei den Modellen zu trawlSpeed konnte der k-NN Classifier am besten abschneiden, jedoch konnte er nicht einen F_1-Score von 85% erreichen. Ein Modell des Support Vector Classifiers konnte bei stopped ebenfalls den gewünschten Wert übertreffen. Die Modelle des Support Vector Classifiers und des Diskriminanzanalyse Verfahrens konnten in allen anderen Fällen jedoch nicht den gewünschten F_1-Score erreichen.

6 Fazit

In dieser Arbeit wurden Machine Learning Algorithmen aus den überwachten Klassifikationsalgorithmen zusammen mit Konzepten aus der Zeitreihenanalyse verwendet, um das Potenzial von Machine Learning Algorithmen in der maritimen Domäne zu ermitteln.

Aus den maritimen Aktivitätsmustern wurden die gewählt, die etwas über die aktuelle Geschwindigkeit eines Schiffs aussagen und verschiedene Classifier wurden genutzt, um Modelle zu lernen, die eine solche Aktivität in einem Datensatz erkennen können. Es hat sich ergeben, dass die Algorithmen durch eine geeignete Vorbereitung der Datensätze, welche sich vor allem auf die von den Definitionen der Aktivitäten und Ereignisse der Domänenexperten stützt, gute Ergebnisse erzielen. Dabei schneiden die Modelle zum Erkennen der punktuellen Ereignisse besser ab als die der Durativen. Die Ergebnisse deuten darauf hin, dass die Algorithmen nicht in der Lage sind zu erkennen, dass eine durative Aktivität durch zwei Ereignisse beschreiben werden kann, die ihren Start- und Endpunkt markieren. Diese Annahme kann jedoch nicht überprüft werden, da die Modelle einer Black-Box gleichen und nicht eingesehen werden kann, was gelernt wurde.

In den Tests haben der Random Forest Classifier und der k-Nearest Neighbour Classifier die besten Ergebnisse erzielen können. Jedoch konnten in dieser Arbeit nur vier verschiedene Classifier betrachtet werden. Einige der Supervised Learning Algorithmen scheinen in der maritimen Domäne für Ereignisse, die nicht von einer großen Menge an Features abhängen, gut geeignet zu sein.

Alternativ hätten aber auch andere Machine Learning Modelle aus dem Unsupervised Learning verwendet oder weitere Konzepte aus der Zeitreihenanalyse eingebaut werden können.

Mit genügend Wissen über die Domäne und die Funktionsweise der einzelnen Algorithmen ist es wahrscheinlich möglich, Algorithmen auszuwählen oder so zu erweitern, dass sie besser in der maritimen Domäne abschneiden. Dadurch könnten dann auch Modelle gelernt werden, die Aktivitätsmuster erkennen können, die mehr als nur von der Geschwindigkeit und dem Typ eines Schiffs abhängen.

Literaturverzeichnis

[AZ21] A. Artikis and D. Zissis. *Guide to Maritime Informatics*. Springer International Publishing, 2021.

[BD15] R. Bruns and J. Dunkel. *Complex Event Processing: Komplexe Analyse von massiven Datenströmen mit CEP*. essentials. Springer Fachmedien Wiesbaden, 2015.

[CCK+00] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000.

[Kub21] M. Kubat. *An Introduction to Machine Learning*. Springer International Publishing, 2021.

[Laz20] F. Lazzeri. *Machine Learning for Time Series Forecasting with Python*. Wiley, 2020.

[mar] MarineTraffic - A day in numbers - MarineTraffic Blog — marinetraffic.com. `https://www.marinetraffic.com/blog/a-day-in-numbers/`. [Accessed 07-08-2023].

[MP18] R.F. Mello and M.A. Ponti. *Machine Learning: A Practical Approach on the Statistical Learning Theory*. Springer International Publishing, 2018.

[NAA+22] Emmanouil Ntoulias, Elias Alevizos, Alexander Artikis, Charilaos Akasiadis, and Athanasios Koumparos. Online fleet monitoring with scalable event recognition and forecasting. *Geoinformatica*, 26(4):613–644, oct 2022.

[PA19] Manolis Pitsikalis and Alexander Artikis. Composite maritime events, February 2019.

[PAA+16] Kostas Patroumpas, Elias Alevizos, Alexander Artikis, Marios Vodas, Nikos Pelekis, and Yannis Theodoridis. Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427, aug 2016.

[PAD+19] Manolis Pitsikalis, Alexander Artikis, Richard Dreo, Cyril Ray, Elena Camossi, and Anne-Laure Jousselme. Composite event recognition for maritime monitoring, 2019.

[PK18] Chondrodima E. Georgiou H. Petrou P. Tampakis P. Sideridis S. Pelekis N. Theodoridis Y. Patroumpas K., Spirelis D. Final dataset of Trajectory Synopses over AIS kinematic messages in Brest area (ver. 0.8), March 2018. Patroumpas K., Alevizos E., Artikis A., Vodas M., Pelekis N. and Theodoridis Y. Online Event Recognition from Moving Vessel Trajectories. GeoInformatica 21(2): 389-427, 2017.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RDC+19] Cyril Ray, Richard Dréo, Elena Camossi, Anne-Laure Jousselme, and Clément Iphar. Heterogeneous integrated dataset for maritime intelligence, surveillance, and reconnaissance. *Data in Brief*, 25:104141, 2019.

[RDCJ18] Cyril Ray, Richard Dreo, Elena Camossi, and Anne-Laure Jousselme. Heterogeneous Integrated Dataset for Maritime Intelligence, Surveillance, and Reconnaissance, February 2018.

[WM10] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[ZL21] Z.H. Zhou and S. Liu. *Machine Learning*. Springer Nature Singapore, 2021.