

Klassifikation von Tweets zur Erkennung von Trollen

Robin Kösters

Bachelorarbeit

Beginn der Arbeit:	18. September 2020
Abgabe der Arbeit:	18. Dezember 2020
Gutachter:	Prof. Dr. Stefan Conrad Prof. Dr. Martin Mauve

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 18. Dezember 2020

Robin Kösters

Zusammenfassung

Hier kommt eine ca. einseitige Zusammenfassung der Arbeit rein.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Datensätze	2
2	Grundlagen des Text Mining	4
2.1	Preprocessing	4
2.2	Feature Extraction	5
2.3	Dimensionalitätsreduktion	6
3	Klassifikationsverfahren	7
3.1	k-Nearest-Neighbor-Algorithmus	7
3.2	Naiver Bayes-Klassifikator	8
3.3	Support Vector Machine	9
3.4	Entscheidungsbäume	10
3.5	Mehrschichtiges Perzeptron	11
4	Gütekriterien	13
4.1	Konfusionsmatrix	13
4.2	Treffergenauigkeit	14
4.3	Sensitivität und Spezifität	14
4.4	Relevanz und Segreganz	15
4.5	F-Score	15
5	Methodik	16
5.1	Hyperparameteroptimierung	16
5.2	Kreuzvalidierung	16
5.3	Erklärte Varianz	17
5.4	Implementierung	18
6	Evaluation	19
6.1	Hyperparameteroptimierungen	19
6.2	Verfahren im Vergleich	24
7	Abschließendes	25

7.1	Fazit	25
7.2	Ausblick	25
	Literatur	26
	Abbildungsverzeichnis	28
	Tabellenverzeichnis	28

1 Einleitung

1.1 Motivation

Die US-Präsidentschaftswahl 2016 gilt vielen politischen Beobachtern als eindrucksvollstes Beispiel dafür, wie von staatlicher Seite organisierte Desinformationskampagnen den politischen Diskurs nachhaltig verzerren können. Bereits zwei Jahre nach der Wahl kamen britische Wissenschaftler in Zusammenarbeit mit der Firma Graphika zu der Erkenntnis, dass ein Unternehmen mit dem Namen „Internet Research Agency“ (IRA), welches dem russischen Staat nahesteht, im großen Stil versucht hat, amerikanische Wähler via Fake-Postings in sozialen Netzwerken zu beeinflussen. In ihrem Bericht geben die Forscher an, dass mehr als 30 Millionen Benutzer im Zeitraum von 2015 bis 2017 in Berührung mit von der IRA erstellten Inhalten gekommen sind. Die Urheber jener destruktiver Inhalte werden Trolle genannt. (Howard et al., 2019)

In Twitter, welches das hauptsächlich betrachtete soziale Netzwerk in dieser Arbeit sein soll, machen Trolle besonders von der Hashtag-Setzung Gebrauch. Bei einem *Hashtag* handelt es sich um eine durch den Ersteller eines Beitrags vorgenommene Themenfestlegung bzw. -klassifizierung. Eine nicht durch Leerzeichen unterbrochene Zeichenkette wird durch Voranstellen eines Rautezeichens (engl. *hash*) zu einem Hashtag. In der Folge kann ein *Tweet* (plattformeigene Bezeichnung für einen Beitrag) durch das Benutzen der allgemeinen Suchfunktion gefunden werden. Abbildung 1 zeigt einen beispielhaften Tweet der IRA, welcher dem der Arbeit zugrundeliegenden Datensatz entnommen wurde und die Hashtag-Nutzung durch einen Troll illustriert.

Today is the dawn of the trumpreich #MAGA #TrumpForPresident

Abbildung 1: Beispiel eines IRA-Tweets

Kommt es innerhalb eines gewissen Zeitraums zur gehäuften Benutzung eines bestimmten Hashtags, so wird dieser in den „Twitter-Trends“, eine Art Ranking der momentan meistgenutzten Hashtags, aufgeführt. Dies macht die Aktualität bzw. Relevanz eines gesellschaftlichen Themas direkt ablesbar. Für Trolle bietet dies aber die Möglichkeit durch geschickte zeitliche Abstimmung bestimmte Hashtags zu Trend-Hashtags zu machen und den Nutzern so aktiv die Relevanz bestimmter Themen vorzutäuschen.

Insgesamt sind mit den genannten Methoden verschiedenste Arten der Manipulation seitens Trollen denkbar. So können Trolle eine politische Partei oder einen Kandidaten unterstützen und versuchen, ihren Mitbewerbern zu schaden. Beispiele dafür gibt es auch in Deutschland. So gelang es dem ultrarechten Netzwerk „Reconquista Germanica“ während des TV-Duells zwischen Bundeskanzlerin Merkel und ihrem Herausforderer Martin Schulz im Jahr 2017 reichweitenstarke Tweets zu erstellen, welche Politiker diskreditieren (Gensing und Kampf, 2020). Ein weitere tragende Säule von Desinformationskampagnen, welche meist in Verbindung zu Trollen steht, sind die *Fake News*. Hierbei handelt es sich um Falschmeldungen bzw. Nachrichten ohne Wahrheitsgehalt, welche verbreitet werden, um die Öffentlichkeit zu manipulieren. Trolle treten hier in der Regel wechselweise als Urheber und Multiplikatoren solcher Meldungen auf.

Alle bisher beschriebenen Strategien sind gemeinsam als integrative Gesamtstrategie da-

zu geeignet, um die Ausgänge demokratischer Wahlen zu beeinflussen. Die Initiative „ichbinhier e.V.“ und das Londoner „Institute for Strategic Dialogue“ (ISD) kamen in einer Studie beispielsweise zu der Schlüsselerkenntnis, dass rechtsextreme Trollnetzwerke im Bundestagswahlkampf 2017 Urheber einer ausgedehnten und erfolgreichen „pro-AfD-Wahlkampagne“ waren (Kreißel et al., 2018). Das Ziel der Desinformationskampagnen, gleich ob aus dem Inland oder Ausland gesteuert, ist die Destabilisierung der demokratischen Institutionen bzw. der Demokratie als Ganzes. Aus diesem Grund möchte ich in dieser Arbeit meinen Teil dazu beitragen, dass eine Lösung für dieses Problem gefunden wird.

1.2 Datensätze

Im Rahmen dieser Arbeit soll ein Verfahren entwickelt werden, welches die zuverlässige Erkennung von Troll-Inhalten auf Twitter ermöglicht. Hierbei kommen verschiedenste Techniken der Textklassifikation zum Einsatz. Diese werden auf zwei Datensätze angewandt, auf deren Eigenschaften und Ursprünge an dieser Stelle eingegangen werden soll. Die Grundmenge des ersten Datensatzes ist eine von Linvill und Warren (2018) veröffentlichte Sammlung von rund 3 Millionen Tweets der IRA. Hier wurden die Tweets mit den zehn häufigsten Hashtags entnommen.

Beim zweiten Datensatz handelt es sich um im Vorfeld eigens extrahierte Tweets von echten Profilen. Um eine thematische Ähnlichkeit herzustellen, wurden nur Tweets mit den gleichen Hashtags wie im ersten Datensatz ausgewählt. Dies ist entscheidend, da eine thematische Ferne gleichbedeutend mit einer zu offensichtlichen Unterscheidbarkeit ist. Die Ergebnisse der Klassifikation wären in diesem Fall erwartbar klar und würden damit einen geminderten praktischen Nutzen nach sich ziehen. Es ist in diesem Zusammenhang auch nicht unwichtig, dass Trolle sich oftmals als authentische Benutzer tarnen und in deren Diskussionsräume vordringen. Die Ergebnisse sind erst dann von Wert, wenn auch diese, in der Theorie schwieriger zu unterscheidenden User, gut unterschieden werden. Aus jenen genannten Gründen ist die thematische Nähe beider Datensätze von entscheidender Bedeutung. Weiterhin wurde darauf geachtet, dass die Hashtags ungefähr gleich unter den Tweets verteilt sind. Tabelle 1 zeigt den prozentualen Anteil eines jeden Hashtags am Gesamtdatensatz. Hier sind Differenzen von höchstens 5%, meist aber um die 1% zu beobachten.

Zur Sicherheit wurde am Ende der Extraktion die Mengendifferenz beider Datensätze gebildet um die Datensätze streng disjunkt zu halten.

Hashtag	Troll	Nichttroll	Hashtag	Troll	Nichttroll
#news	42,50%	40,50%	#topNews	05,00%	05,87%
#sports	16,06%	15,99%	#MAGA	05,07%	05,23%
#politics	13,12%	15,25%	#BlackLivesMatter	03,96%	04,56%
#world	09,13%	13,94%	#health	03,84%	04,60%
#local	08,62%	08,83%	#tcot	03,99%	05,93%

Tabelle 1: Verteilung der Hashtags

Der neu entstehende zusammengesetzte Datensatz besteht aus knapp 630.000 Tweets. Tabelle 2 zeigt zum Vergleich die wichtige Kennzahlen beider Bestandteile wie den Erstellungszeitraum und die durchschnittliche Länge der Zeichenketten.

Eigenschaft	Troll	Nichttroll
Anzahl Tweets	303.036	324.873
unterschiedliche Autoren	770	68.706
Zeitraum	01/2015 - 09/2017	01/2015 - 05/2018
durchschnittliche Länge	78,64 Zeichen	122,27 Zeichen

Tabelle 2: Vergleich der Datensätze

2 Grundlagen des Text Mining

Im nachfolgenden Kapitel werden jene Konzepte des *Text Minings* erläutert, welche grundlegend für das Verständnis von Methoden der Textklassifikation sind. Abbildung 2 zeigt die Phasen, die beginnend beim Textkorpus (Sammlung der Ausgangstexte) bis zur abschließenden Einstufung durchlaufen werden. Hieran werde ich mich bei den Ausführungen orientieren.

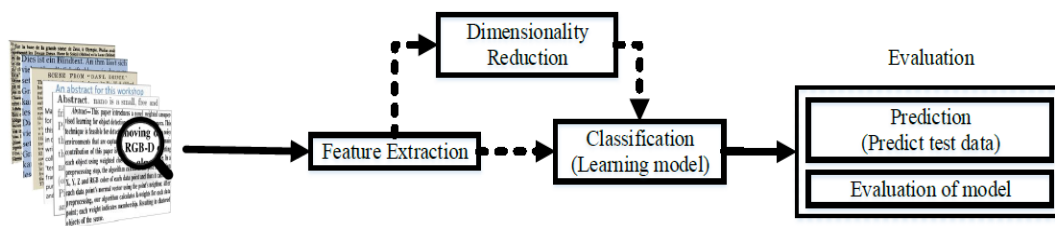


Abbildung 2: Pipeline der Textklassifikation (Kowsari et al., 2019)

2.1 Preprocessing

Der rohe, unbehandelte Text wird beim *Preprocessing* (deutsch: Vorbehandlung) in eine Form überführt, die eine nachfolgende Repräsentation der Daten durch Vektoren und ähnliche Methoden ermöglicht. Je nach gewünschter Anwendung sind hier verschiedene Vorverarbeitungsverfahren in Betracht zu ziehen.

Der Begriff Wortsegmentierung bzw. **Tokenisierung** bezeichnet die Zerteilung des Ausgangstexts in kleinere Einheiten, sogenannte *Token* (Manning und Schütze, 1999). In der Regel handelt es sich hierbei um einzelne Wörter, Zahlen und Satzzeichen. Als grundsätzliches Abgrenzungszeichen gelten in den meisten Sprachen die *Whitespace*-Zeichen (Leerzeichen, Tab, Newline-Zeichen). Dies wird auch auf die zu analysierenden Tweets zutreffen, da diese fast ausschließlich in englischer Sprache verfasst sind.

Er	rief	:	„	Baum	fällt	!	“
----	------	---	---	------	-------	---	---

Abbildung 3: Tokenisierung eines Beispielsatzes

Eine weitere Methode der Vorverarbeitung ist das Entfernen von **Stoppwörtern**. Dies sind sehr häufig auftretende Wörter wie „der“, „die“, „das“, „und“ oder „von“, welche vornehmlich eine grammatikalische Funktion und keine Bedeutung im Sinne von Semantik haben. Ob die Filterung von Stoppwörtern angezeigt ist, hängt vom Anwendungsfall ab. Ein grundsätzlich denkbarer Fall wäre, dass Trolle in ihren Tweets entweder über- oder unterdurchschnittlich Gebrauch von diesen Wörtern machen. Eine

Filterung würde dann zu einem Verlust von Unterscheidungsmerkmalen und damit zu einer erschwerten Klassifikation führen. Im Falle dass kein Unterschied zwischen Trollen und Nichttrollen in dieser Eigenschaft besteht, bringt die Filterung durch weniger zu berücksichtigende Token Performanceverbesserungen mit sich.

2.2 Feature Extraction

Bevor eine Klassifikation vorgenommen werden kann, müssen aus den vorliegenden Texten Merkmale gewonnen werden, welche einen Vergleich im Bezug auf Ähnlichkeit und Unterschiedlichkeit ermöglichen. Die extrahierten Merkmale bilden dann zusammengefasst in einem Merkmalvektor eine eindeutige numerische Repräsentation eines Textes. Die Vektorisierung schafft zudem die Grundlage für verschiedenste Operationen wie Abstandsmessungen, die später bei den Klassifikationsverfahren zum Einsatz kommen werden.

Die einfachste Vektorisierungstechnik ist die **Bag-of-Words** (BoW) (Ramos, 2003). Hier wird ein Text durch einen Vektor mit den Begriffshäufigkeiten (auch: *Term Frequencies* (TF)) aller zuvor extrahierten Tokens des Textkorpus repräsentiert. Bei einem Beispieldatensatz mit den beiden Texten „my coffee is too hot“ und „my tea is too cold“ und dem zuvor extrahierten Vokabular

{ "my", "coffee", "hot", "tea", "cold" }

ergibt sich die folgende Repräsentation:

$$\begin{aligned} v_1 &= [1, 1, 1, 0, 0] \\ v_2 &= [1, 0, 0, 1, 1] \end{aligned}$$

Eine Erweiterung dieser TF-Methode ist *Term Frequency-Inverse Document Frequency* (TF-IDF) (Ramos, 2003). Hier wird die Vorkommenshäufigkeit anders gewichtet, um dem Aspekt gerecht zu werden, dass einige Begriffe überproportional oft vorkommen. Gleichung 1 zeigt die Gewichtung

$$w(d, t) = TF(d, t) \cdot \log \left(\frac{N}{DF(t)} \right) \quad (1)$$

wobei N die Anzahl der Texte im Korpus und $DF(t)$ die Anzahl der Texte, welche den Begriff t enthalten, ist.

Eine Möglichkeit, Wortkombinationen bzw. bestimmte Formulierungen als Merkmal zu berücksichtigen ist das **N-Gramm** (Kowsari et al., 2019). Dies ist die Zusammenfassung von N aufeinanderfolgenden Token in einem Text. Folglich handelt es sich bei den Elementen einer Bag of Words um 1-Gramme. Das nachfolgende Beispiel zeigt die Repräsentation des Textes „Hier sehen Sie ein Beispiel.“ mit 2-Grammen:

{ „Hier sehen“, „sehen Sie“, „Sie ein“, „ein Beispiel“ }

2.3 Dimensionalitätsreduktion

Bei sehr umfangreichen Datensätzen wie den mehr als 600.000 Tweets in dieser Arbeit werden in der Anwendung der zuvor beschriebenen Verfahren meist hochdimensionale Vektoren erzeugt. In der Folge werden viele Operationen bei späteren Algorithmen der Textklassifikation eine hohe Zeit- und Speicherkomplexität besitzen. Diesem Effekt versucht man im Voraus durch Dimensionalitätsreduktion entgegenzuwirken.

Ein erstes verwendetes Verfahren ist die Hauptkomponentenanalyse bzw. **Principal Component Analysis** (PCA) (Jolliffe, 2002). Mit diesem ist es möglich, in der Punktwolke der vorhandenen Vektoren all jene Vektor-Komponenten herauszufinden, die für die größte Varianz verantwortlich sind, also den größten Informationsgehalt haben. Der mathematische Mechanismus dahinter ist die Hauptachsentransformation: Es wird eine Ladungsmatrix aus den Eigenvektoren der Kovarianzmatrix gebildet, aus welcher der Anteil der Varianz jeder Komponente an der Gesamtvarianz ersichtlich ist. In der Folge können Komponenten, welche wenig Varianz beitragen, ohne nennenswerten Informationsverlust verworfen werden.

Eine mit PCA verwandte Methode ist die Singulärwertzerlegung (engl. *singular value decomposition*, SVD) (Baker, 2013). Der Hauptunterschied ist hier, dass die Komponentenanalyse direkt auf den Merkmalvektoren und nicht auf der Kovarianzmatrix ausgeführt wird. Am Ende erhält man eine Faktorisierung (Gleichung 2) bestehend aus drei Matrizen.

$$M = U \cdot \Sigma \cdot V^* \quad (2)$$

Die für die Reduktion entscheidende Matrix Σ (siehe Beispiel in Gleichung 3) enthält auf ihrer Diagonalen die sogenannten Singulärwerte (rot). Falls eine Dimensionsreduktion möglich ist, sind Spaltenvektoren bestehend aus Nullen (blau) als letzte Zeilen in Σ vorhanden.

$$\Sigma = \begin{pmatrix} \textcolor{red}{2} & 0 & 0 & \textcolor{blue}{0} \\ 0 & \textcolor{red}{1} & 0 & \textcolor{blue}{0} \\ 0 & 0 & \textcolor{red}{1} & \textcolor{blue}{0} \\ 0 & 0 & 0 & \textcolor{blue}{0} \end{pmatrix} \quad (3)$$

3 Klassifikationsverfahren

Nachdem die Tweets vorbearbeitet, die Merkmale gewonnen und die Dimensionen zwecks verbesserter Performance reduziert wurden, können sie nun als Trainingsdatensatz für ein Klassifikationsverfahren verwendet werden. Im nachfolgenden Kapitel werden verschiedene solcher Verfahren vorgestellt und im Hinblick auf Voraussetzungen, Stärken und Schwächen analysiert. Ein besonderes Augenmerk soll in der Analyse auf jenen Parametern liegen, welche die Ergebnisse beim vorliegenden Datensatz maßgeblich beeinflussen können.

3.1 k-Nearest-Neighbor-Algorithmus

Der k-Nearest-Neighbor-Algorithmus (kNN) ist die erste hier behandelte Methode der Klassifikation (Guo et al., 2004). Der zu klassifizierende Text wird zunächst analog zu den Trainingstexten vektorisiert. Über ein geeignetes Abstandsmaß werden nun die k räumlich nächsten Nachbarn bestimmt. Der Text wird nun der Klasse zugeordnet, der die Mehrheit der Nachbarn angehören. Abbildung 4 zeigt die k-Nearest-Neighbor-Klassifikation eines Punktes x für $k = 3$. Da die meisten Nachbarn hier dem Troll-Datensatz angehören, würde der zu x gehörige Tweet somit als Troll-Tweet klassifiziert werden.

Die Wahl von k ist entscheidend für die Qualität des Ergebnisses. Entscheidet man sich beispielsweise für ein zu kleines k , so ist möglich, dass vereinzelte Ausreißer die Genauigkeit trüben. Ist es auf der anderen Seite zu groß, so werden wahrscheinlich zu weit entfernte Punkte bei der Klassifikation miteinbezogen, was das Ergebnis wiederum verfälschen kann. Ferner ist bei Vorhandensein von 2 Klassen ein ungerades k zu wählen, da andernfalls ein Unentschieden möglich ist.

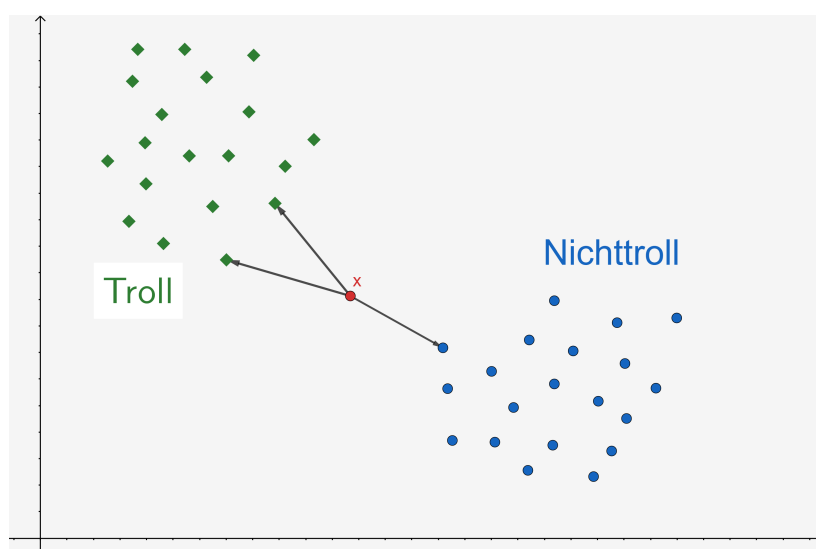


Abbildung 4: KNN-Klassifizierung für $k = 3$

Eine zwingende Voraussetzung, um zuverlässig mit dem KNN-Algorithmus in diesem Projekt arbeiten zu können, ist die Dimensionalitätsreduktion. Die Merkmalsextraktion bringt bei den mehr als 600.000 Tweets hochdimensionale Vektoren hervor. Unbehandelt wären aufgrund der komponentenweisen Abstandsmessung Laufzeiten von einigen Minuten zu erwarten.

Die Vorteile dieses Verfahrens sind die einfache Implementierung und seine Eignung für alle möglichen Ausprägungen von Merkmalsräumen. Als Schwächen werden die bereits angesprochenen Probleme mit der Performance angesehen.

3.2 Naiver Bayes-Klassifikator

Der Naive Bayes-Klassifikator (NB) ist ein statistisches Verfahren der Klassifikation (Rish, 2001). Die Grundlage der Berechnung bildet hier der Satz von Bayes, bekannt aus der Wahrscheinlichkeitstheorie. In seiner herkömmlichen Interpretation beschreibt dieser die Berechnung der Wahrscheinlichkeit, dass ein Ereignis dem anderen vorausgegangen ist. Wendet man dies auf einen gegebenen Text $t \in T$ und die Klasse $k_i \in K$ an, so erhält man die Formel für die Wahrscheinlichkeit, dass t der Klasse k_i angehört (siehe Gleichung 4).

$$P(k_i|t) = \frac{P(t|k_i) \cdot P(k_i)}{P(t)} \quad (4)$$

Der Klassifikator bestimmt nun diejenige Klasse k_i , für die der Wert dieser Formel maximal ist. Mathematisch formuliert:

$$k = \arg \max_{k_i \in K} P(k_i|t) = \arg \max_{k_i \in K} \frac{P(t|k_i) \cdot P(k_i)}{P(t)} \quad (5)$$

Die Werte für $P(k_i)$ und $P(t)$ werden a priori über relative Anteilshäufigkeiten, also über den Anteil einer Klasse und den Anteil eines Textes am gesamten Merkmalraum, bestimmt. Für die Maximierung wird meist die Maximum-Likelihood-Methode benutzt. Jeder Klasse wird vor der Klassifizierung eine bestimmte Form der Wahrscheinlichkeitsverteilung, meist eine Normalverteilung, und die stochastische Unabhängigkeit der Merkmale unterstellt. Letztere Annahme ist „naiv“ weil nicht immer zutreffend, der Algorithmus liefert in der Praxis bei vielen Anwendungen dennoch solide Ergebnisse. Wird diese Annahme hingegen durch stark korrelierte Daten gröber verletzt, so sind Genauigkeitsverluste zu erwarten, was eine Schwäche dieses Verfahrens ist. Im Rahmen der Evaluation werden Ergebnisse unter Annahme einer Normalverteilung und einer Multinomialverteilung verglichen.

3.3 Support Vector Machine

Ein anderes Lernmodell trägt den Namen Support Vector Machine (SVM) (Nayak et al., 2015). Die Grundidee ist hier, die vorliegenden Klassen im multidimensionalen Raum durch Einschub einer Hyperebene räumlich zu trennen. Die Ebene wird dabei so platziert, dass der Abstand zu den nächstliegenden Vektoren der beiden Klassen, den sogenannten Stützvektoren (engl. *support vectors*) maximal ist. Abbildung 5 zeigt beispielhaft zwei solcher durch eine Hyperebene (rote Gerade) getrennte Klassen in einem zweidimensionalen Merkmalraum.

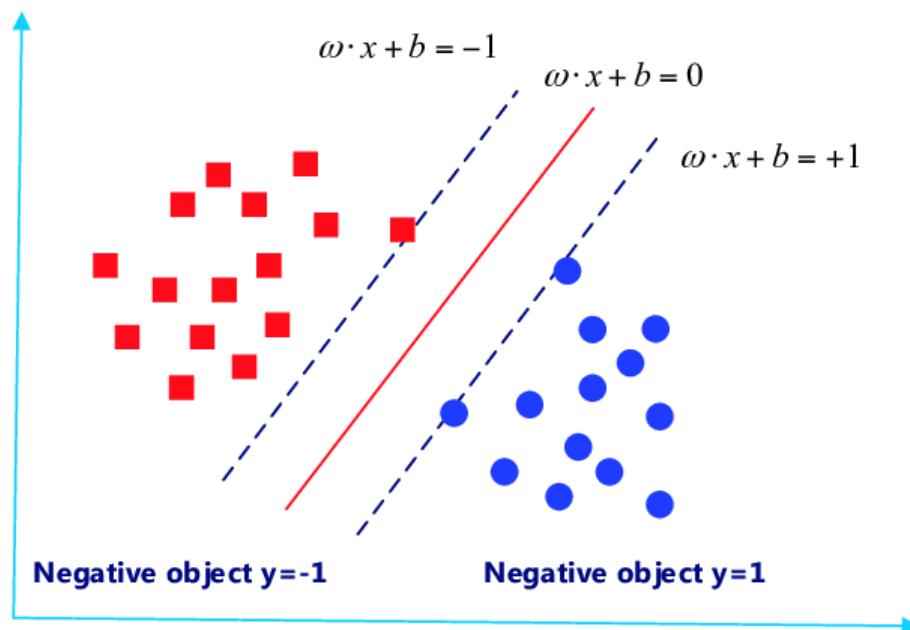


Abbildung 5: SVM-Klassifizierung (Zhou et al., 2016)

Für ein zu klassifizierendes Objekt wird nun mithilfe einer Entscheidungsfunktion festgestellt, auf welcher Seite der Ebene es liegt und kann so eindeutig einer Klasse zugeordnet werden. Ist der Einschub einer Hyperebene nicht möglich, d.h. sind die Klassen nicht linear trennbar, so behilft man sich mit dem sogenannten „Kernel-Trick“: Der Merkmalraum wird auf einen höherdimensionalen Raum abgebildet, in welchem die Klassen dann linear trennbar sind. Diese Transformation benötigt eine hohe Rechenleistung und stellt deshalb bei stark überlappenden Klassen eine Schwäche dieses Verfahrens dar. Bei weniger überlappenden oder linear trennbaren Klassen ist der Algorithmus in der Einstufung jedoch sehr schnell. Ein weiterer Vorteil ist der relativ geringe Speicherverbrauch, da zur Klassifikation nur ein Teil der Trainingsdaten gebraucht wird.

Zwei Parameter, die die Ergebnisse beeinflussen können, sind zum einen der Bestrafungsparameter C und der Gamma-Wert. C gibt an, wie stark die Gefahr der Falschklassifizierung berücksichtigt werden soll. Je niedriger der Wert, desto leichter wird es, eine Trennung zu vollziehen und dabei einzelne Ausreißer zu ignorieren. Allerdings findet dann bei mäßiger bis starker Überlappung keine eindeutige Trennung der Klassen statt.

Je höher der Wert, desto strenger versucht der Algorithmus, alle Punkte in der Entscheidung miteinzubeziehen. Hier besteht die Gefahr der Überanpassung. Der Gamma-Wert gibt an, wie stark einzelne Punkte beim Kernel-Trick Einfluss auf die Gruppierung haben, also wie granular die Gruppen sind. Bei linear trennbaren Klassen kommt er demnach nicht zur Anwendung. Um zu guten Ergebnissen zu kommen, ist eine gute Balance der beiden Werte, in jedem Fall aber die Anpassung an den Datensatz von Nöten.

3.4 Entscheidungsbäume

Die Grundlage für ein weiteres Klassifikationsverfahren sind die sogenannten Entscheidungsbäume (engl. *decision trees*) (Rokach und Maimon, 2005). Hierbei handelt es sich um herkömmliche, aus der Graphentheorie bekannte Bäume, welche eine Entscheidungsfolge abbilden. Für jedes zu klassifizierende Objekt wird der Baum einmal durchlaufen. An jedem inneren Knoten wird nun der Wert eines extrahierten Merkmals abgefragt. Je nach Ergebnis des Vergleichs wird entschieden, welches Kind als nächstes besucht wird. Die Blätter beinhalten die möglichen Klassen, sodass nach der letzten Entscheidung mit Ankommen am Ende des Baumes die Klassifikation vollzogen ist. Abbildung 6 zeigt beispielhaft einen Entscheidungsbaum für die Klassifikation eines Messestandorts anhand von drei Merkmalen. Die möglichen Klassen für einen Standort sind „geeignet“ und „ungeeignet“.

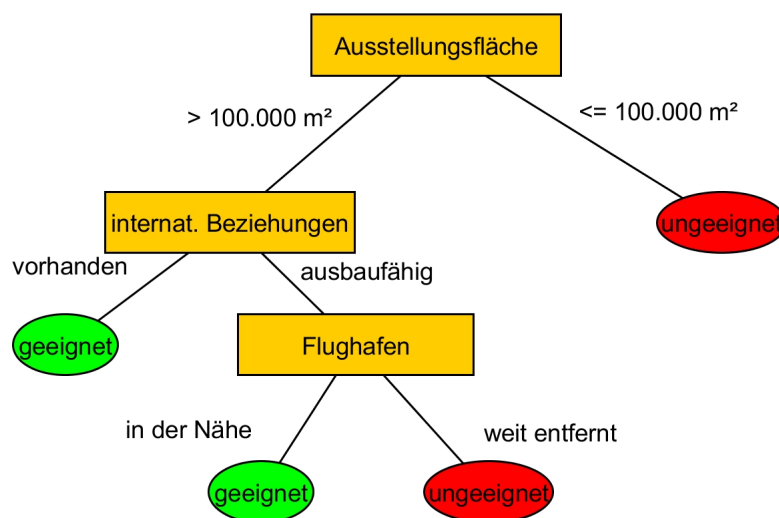


Abbildung 6: Entscheidungsbaum zur Klassifizierung eines Messestandorts

Das Training bzw. das Lernen besteht bei diesem Verfahren im Aufbau des Baumes. Beginnend bei der Wurzel wird nun dasjenige Merkmal bestimmt, welches den größten Fortschritt in der Klassifikationsentscheidung bringt. Ein gebräuchliches Maß dafür ist die *Gini Impurity*, welche die Homogenität bzw. Heterogenität der Daten anzeigt. Gleichsam anwendbar ist hier auch die Entropie, welche den mittleren Informationsgehalt des

Merkmalvektors misst. Das gefundene Merkmal dient nun zur Aufteilung der ursprünglichen Datenmenge. Auf die neu entstehenden Mengen wird diese Methode nun rekursiv angewendet bis Mengen erzeugt werden, deren Elemente alle der gleichen Klasse angehören.

Durch die klare Strukturierung und grafische Darstellbarkeit ist die Entscheidungsbaum-Klassifikation als Konzept leicht verständlich. Ein weiterer Vorteil ist eine automatische Merkmalfilterung beim Aufbau des Baums: Merkmale, die durch Gini Impurity bzw. Entropie als irrelevant erkannt wurden, können leicht weggelassen werden. Hier sind Zeit- und Speichergewinne möglich.

Eine Schwäche dieses Verfahrens ist die schwache Robustheit der Entscheidungsbäume. Dies bedeutet, dass kleine Änderungen im Ausgangsdatensatz zu sehr großen Veränderungen in der Datenstruktur und der anschließenden Klassifikation führen können.

3.5 Mehrschichtiges Perzeptron

Das mehrschichtige Perzeptron (engl. *multi-layer perceptron*, *MLP*) ist ein Vertreter der künstlichen neuronalen Netze (Du und Swamy, 2014). In seinem grundlegenden Aufbau ähnelt es einem Logikgatter. Die grundlegenden Bausteine, analog zu Gehirnzellen Neuronen genannt, sind in Schichten angeordnet. Die Eingabeschicht nimmt den Merkmalvektor komponentenweise entgegen, während die Neuronen der Ausgabeschicht das Klassifikationsergebnis ausgeben. Die dazwischenliegenden Schichten werden versteckte Schichten bzw. *hidden layers* genannt. Abbildung 7 illustriert ein Beispiel für Netz, welches einen vierdimensionalen Vektor verarbeitet.

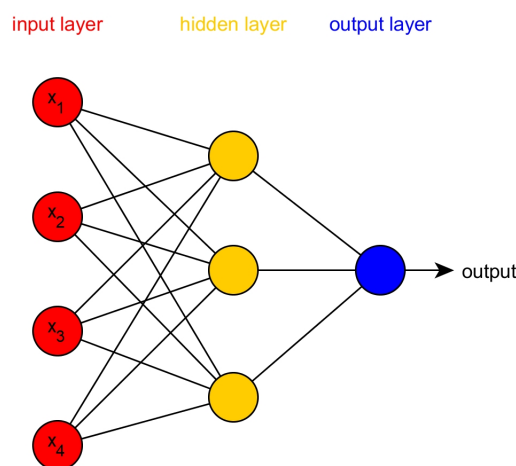


Abbildung 7: Beispiel für mehrschichtiges Perzeptron

Jedes Neuron einer Schicht hat eine Verbindung, ähnlich einer Kante, zu einem Neuron (oft auch zu allen) der nächsten Schicht. Die Verbindung trägt ein Gewicht. Dazu trägt auch jedes Neuron einer versteckten Schicht und der Output-Schicht ein sogenanntes Verzerrungsgewicht bzw. *bias weight* b .

Der Lernprozess läuft nun wie folgt ab: Beginnend bei den Input-Neuronen wird für jede

Eingabe eine gegebene Aktivierungsfunktion berechnet. Das Ergebnis wird an die nachfolgenden Neuronen weitergeschickt und erneut die Aktivierungsfunktion berechnet. Ist es schließlich in der Output-Schicht angekommen, so wird das Ergebnis interpretiert und mit der vorher deklarierten Klasse verglichen. Ist das Ergebnis falsch, so werden die Gewichte des Netzes von hinten nach vorne neujustiert. Man spricht hier von *Backpropagation*. Auf diese Weise lernt das Netz mit jedem Fehler dazu und die Klassifikation wird zunehmend genauer.

Als Aktivierungsfunktion werden klassischerweise drei verschiedene Funktionen herangezogen (Nwankpa et al., 2018). Die erste Funktion ist der *Tangens hyperbolicus* (kurz tanh).

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (6)$$

Oft verwendet wird auch die folgende logistische Funktion

$$f(x) = \frac{1}{1 + e^{-x}} \quad (7)$$

Eine neuere Form der Aktivierungsfunktion ist die *rectified linear unit* (kurz: ReLu). Sie ist folgendermaßen definiert:

$$f(x) = \max(0, x) \quad (8)$$

4 Gütekriterien

Die bereits besprochenen Klassifikationsverfahren werden nach dem „Train-and-Test“-Prinzip auf den vorliegenden Datensatz angewandt. Die Grundmenge der Tweets wird hierbei in eine Trainings- und eine Testmenge aufgeteilt. Die Trainingsmenge dient dem jeweiligen Verfahren als Lerneingabe, während die Testmenge zur Beurteilung der Performance herangezogen wird. Nach der abgeschlossenen Testklassifikation liegen Ergebnisse in Form der den Tweets zugeordneten Labels (Troll oder Nichttroll) vor. An diesem Punkt muss nun geklärt werden, in welcher Weise eine Auswertung vorgenommen werden kann. Wie kann die Qualität eines binären Klassifikators gemessen werden? Was für eine Aussagekraft haben verschiedene Gütekriterien? Welche Nachteile gibt es? Welche Kriterien zeigen die Stärken, welche die Schwächen eines Verfahrens auf? Diese Fragen sollen in diesem Kapitel besprochen werden, ehe die Ergebnisse evaluiert werden können. Dazu werden verschiedene Gütemaße (Hossin und M.N, 2015) vorgestellt. Neben Qualität im Sinne von korrekt oder falsch soll es hier auch um Qualität im Sinne von schnell (Zeitkomplexität) gehen.

4.1 Konfusionsmatrix

Da bei den Tweets der Testmenge im Voraus die richtige Klasse bekannt war, kann für jedes klassifizierte Element eine Korrekt- oder Falschklassifizierung festgestellt werden. Typischerweise wird bei einer binären Klassifikation eine der beiden Klassen als positiv (im Sinne von „bestätigend“) und die andere als negativ aufgefasst. Der gegebene Zweck der Arbeit (Trollerkennung) und die Semantik der beiden Labelbezeichnungen legen in diesem Fall nahe, dass „Troll“ als positiv („Troll gefunden“) und „Nichttroll“ als negativ („keinen Troll gefunden“) aufgefasst wird. Kombiniert ergeben sich hieraus nun folgende Kennzahlen:

- r_p ist die Anzahl richtig positiver, r_n die Anzahl richtig negativer Klassifizierungen.
- f_p bezeichnet die Anzahl falsch positiver, f_n die Zahl falsch negativer Klassifizierungen.

Eine häufige Darstellung von Korrekt- und Falschklassifikation ist die sogenannte Konfusionsmatrix oder Wahrheitsmatrix (siehe Tabelle 3), in welcher die genannten Kennwerte direkt ablesbar sind.

		korrekte Klasse	
Test- ergebnis		Troll ($r_p + f_n$)	Nichttroll ($f_p + r_n$)
	Troll ($r_p + f_p$)	r_p	f_p
	Nichttroll ($f_n + r_n$)	f_n	r_n

Tabelle 3: Konfusionsmatrix der Trollerkennung

Die genannten Werte bilden die Berechnungsgrundlage aller nachfolgend genannten Gütemaße. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit

amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

4.2 Treffergenauigkeit

Die Treffergenauigkeit (engl. **accuracy**) ist das am häufigsten für die Auswertung von Klassifikationsergebnissen gebrauchte Gütemaß. Sie ist folgendermaßen definiert:

$$accuracy = \frac{r_p + r_n}{r_p + f_p + r_n + f_n} = \frac{r_p + r_n}{N} \quad (9)$$

Wie Formel 9 zu entnehmen ist, handelt es sich hierbei um den Anteil der richtig klassifizierten Objekte an der Gesamtzahl N aller Objekte. Deshalb nennt man dieses Maß auch Korrektklassifizierungsrate.

Die Treffergenauigkeit ist von allen Maßen in der Gesamtschau am intuitivsten, da sie die Verlässlichkeit der Klassifikation, also die Wahrscheinlichkeit, dass die getroffene Vorhersage korrekt ist, misst. Für die Trollerkennung ist dies von besonderer Bedeutung: Sowohl das Nichterkennen eines Trolls als auch die „Falschbeschuldigung“ eines echten Benutzers können kritische Konsequenzen nach sich ziehen. Aus diesem Grund ist es wichtig zu wissen, mit welcher Wahrscheinlichkeit eine Verwechslung auszuschließen ist.

4.3 Sensitivität und Spezifität

Die **Sensitivität** (engl. *recall*) ist ähnlich definiert wie die Treffergenauigkeit, mit dem Unterschied, dass hier die positiven Werte (Trollmenge) isoliert betrachtet werden. Man könnte sie aus diesem Grund auch als „Troll-Korrektklassifizierungsrate“ bezeichnen. Sie beschreibt demnach den Anteil der richtig erkannten Troll-Tweets an der Gesamtzahl aller Troll-Tweets.

$$recall = \frac{r_p}{r_p + f_n} \quad (10)$$

Analog dazu ist die **Spezifität** (engl. *specifity*) definiert, sie gibt den Anteil der korrekt als Nichttroll-Tweets erkannten Texte an der Gesamtzahl aller Nichttroll-Tweets an.

$$specifity = \frac{r_n}{r_n + f_p} \quad (11)$$

Die Betrachtung dieser Werte ist vor allem dann von Vorteil, wenn sie sich merkbar voneinander unterscheiden. In diesem Fall liefert die Treffergenauigkeit allein nicht genügend Aufschluss über die Korrektklassifizierung von Trollen und Nichttrollen und es lohnt sich, diese Einzelbetrachtung vorzunehmen.

Auch wenn eine Verwechslung, wie bereits ausgeführt, in jedem Fall problematisch ist, so ist die Falschklassifizierung eines realen Users besonders verheerend, da die getroffenen Gegenmaßnahmen (i.d.R. Sperrungen) unrechtmäßig wären. Aus diesem Grund ist die Spezifität für die Trollerkennung, sofern damit Gegenmaßnahmen einhergehen, von besonderer Bedeutung. Da Trolle speziell in großer Gruppenzahl Wirksamkeit erreichen, ist das fälschliche „Freisprechen“ einiger Trolle weniger kritisch, solange der Rest überwiegend korrekt klassifiziert wird. Dies macht die Sensitivität zweitrangig, aber dennoch

wichtig. Diese Art der Klassifikation, bei der f_p besonders niedrig gehalten werden soll, nennt man **konservative** Klassifikation.

4.4 Relevanz und Segreganz

Die **Relevanz** (engl. *precision*) verfolgt im Vergleich zur Sensitivität einen umgekehrten Ansatz. Während letztere die Frage „Wieviel Prozent der Trolle werden richtig erkannt?“ beantwortet, geht es hier um die Behandlung der Frage „Wieviel Prozent der Troll-Vorhersagen stimmen?“. Definiert ist die Relevanz als Anteil der richtig positiven Vorhersagen an der Gesamtzahl aller positiven Vorhersagen.

$$precision = \frac{r_p}{r_p + f_p} \quad (12)$$

Das negative Äquivalent zur Relevanz ist die **Segreganz** (engl. *negative predictive value*, NPV). Sie beschreibt das Verhältnis von den korrekt negativen Vorhersagen an der Gesamtzahl aller negativen Vorhersagen.

$$NPV = \frac{r_n}{r_n + f_n} \quad (13)$$

Der Fokus bei diesen Maßen liegt nicht auf der Abdeckung der jeweiligen Klassen, sondern auf der Vorhersagegenauigkeit. Es gelten hier ähnliche Implikationen im Bezug auf sich ergebende Probleme wie bei den zuletztgenannten Maßen.

4.5 F-Score

Der **F-Score** kombiniert die bereits beschriebenen Gütemaße Sensitivität und Relevanz und setzt sie mittels des harmonischen Mittels zueinander in Beziehung.

$$F_\alpha = (1 + \alpha^2) \cdot \frac{precision \cdot recall}{\alpha^2 \cdot precision + recall} \quad (14)$$

Formel 14 zeigt die Formel für ein allgemeines α , welches die gewählte Gewichtung beschreibt. Die am häufigsten verwendete Gewichtung ist $\alpha = 1$:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (15)$$

5 Methodik

Das erklärte Ziel dieser Arbeit ist es bekanntermaßen, die beschriebenen Klassifikationsverfahren auf den Datensatz anzuwenden und die Performance mithilfe der Gütemaße aus Kapitel 4 zu messen und auszuwerten. Für eine hinreichende Interpretierbarkeit wird jedoch Kenntnis über das Zustandekommen der Ergebnisse benötigt. Aus diesem Grund soll hier vorbereitend genaustens über die gewählte Vorgehensweise informiert werden.

5.1 Hyperparameteroptimierung

Bei vorausgegangenen Ausführungen wurde angedeutet, dass erwartet wird, dass verschiedene Techniken der Merkmalsextraktion, verschiedene Level der Dimensionalitätsreduktion und verschiedene Hyperparametereinstellungen die Ergebnisse qualitativ beeinflussen werden. Um diese Unterschiede zu messen, wird in dieser Arbeit die sogenannte Hyperparameteroptimierung angewandt. Hierbei wird das gewählte Verfahren wiederholt mit unterschiedlichen, zuvor gewählten Parameterkombinationen auf den Merkmalraum angewandt. Dieses Vorgehen wird **Rastersuche** genannt. Nach erfolgter Ausführung liegen für jede Kombination Punktzahlen vor. Anschließend werden die mittleren Punktzahlen aller Ausprägungen eines Hyperparameters (z.B. von TF und TF-IDF) miteinander verglichen um herauszufinden, welche Einstellungen bei diesem vorliegenden Datensatz zu tendenziell besseren Ergebnissen führen.

5.2 Kreuzvalidierung

Die Klassifikation mit allen genannten Algorithmen des maschinellen Lernens durchläuft stets eine Trainingsphase und eine Testphase. Hierzu wurde der Grunddatensatz vorher in eine Trainings- und eine Testmenge aufgeteilt. Es gibt verschiedene Methoden diese Aufteilung zu organisieren. Die hier gewählte Aufteilungsstrategie für die Hyperparameteroptimierung ist eine **2-fache stratifizierte Kreuzvalidierung** (Abbildung 8).

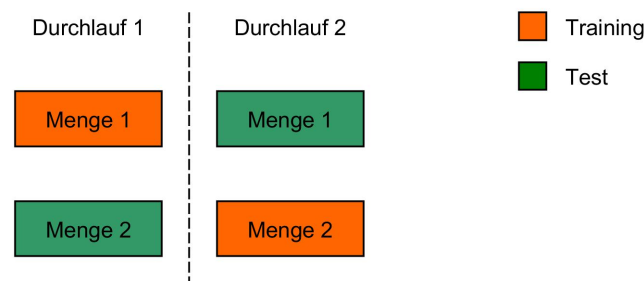


Abbildung 8: Zweifache Kreuzvalidierung

Bei diesem Verfahren wird der Datensatz in zwei gleich große Mengen aufgeteilt. Die Stratifizierung bedeutet in diesem Zusammenhang, dass die Verteilung der Troll-Tweets und Nichttroll-Tweets in den beiden neu entstehenden Mengen genau wiederhergestellt wird. Im Folgenden wird dann in zwei Durchläufen eines Algorithmus wechselseitig jede der zwei Mengen einmal als Trainings- und einmal als Testmenge verwendet. Das Ergebnis für eine Hyperparameterkombination wird dann aus den Mittelwerten der beiden Durchläufe gebildet.

5.3 Erklärte Varianz

Ein Hyperparameter, der grundsätzlich auch in der Rastersuche mit berücksichtigt werden kann, ist das Level der Dimensionalitätsreduktion. Dies bringt jedoch einen fundamentalen Konflikt mit sich: Das Testen von mehreren Werten für die Dimensionen würde gleichsam ein Vielfaches an Ausführungen und damit eine deutlich gesteigerte Laufzeit mit sich bringen. Andererseits liefern mehr Vektorkomponenten auch mehr Informationsgehalt (eine größere Varianz) und damit bessere Ergebnisse, die auch interessant für die Optimierung sind. Letztere Proportionalität ermöglicht aber auch eine alternative Vorgehensweise: Man kann das Reduktionslevel d für alle Ausführungen konstant wählen. In diesem Fall ist es entscheidend, ein d zu wählen, was den Anforderungen der Laufzeit und der Qualität gerecht werden kann. Wie in Kapitel 2.3 beschrieben, kann der Informationsgehalt der einzelnen Vektorkomponenten im Zuge einer Hauptkomponentenanalyse oder einer Singulärwertzerlegung ermittelt werden. Man spricht hier von der durch eine Komponente **erklärten Varianz**.

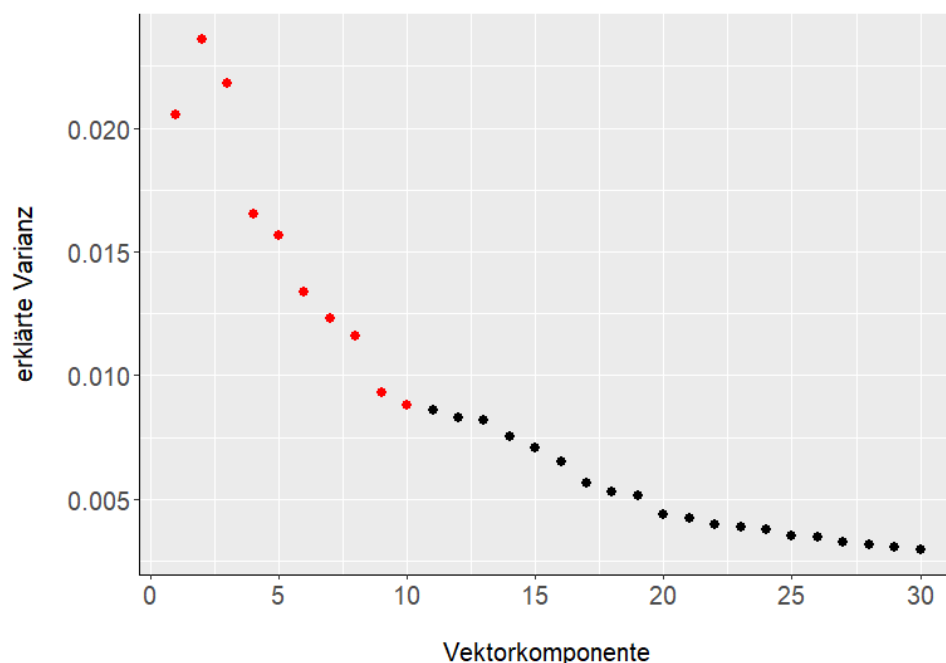


Abbildung 9: Erklärte Varianz der Komponenten

Abbildung 9 zeigt nun die 30 höchsten Werte der erklärten Varianz nach einer Singulärwertzerlegung der Vektoren des Datensatzes. Demnach erklären 3 Komponenten über 2% der Varianz, 5 Komponenten zwischen 2 und 1% der Varianz und der Rest die übrige Varianz. Kumuliert man diese Werte, werden durch die besten 10 Komponenten (rot) 15% der Varianz erklärt. Dies ist für die geringe Anzahl der Komponenten ein vertretbarer Informationsgehalt. Da $d = 10$ gleichzeitig ebenfalls lauffzeitverträglich ist, wird dieser Wert konstant für alle nachfolgenden Anwendungen gewählt.

5.4 Implementierung

Zwecks Nachvollziehbarkeit und Transparenz soll an dieser Stelle kurz auf die parallel erfolgte praktische Anwendung der beschriebenen Techniken eingegangen werden.

Im Rahmen dieser Abschlussarbeit wurde ein Kommandozeilen-Tool namens „TrollDetector“¹ in der Programmiersprache Python entwickelt. Dieses hat mehrere Funktionen. Zum einen ist damit möglich, ein beliebiges Klassifikationsverfahren mit selbst gewählten Hyperparametern auf den Datensatz anzuwenden. Dies geschieht nach dem bereits erwähnten „Train-and-Test“-Verfahren. Hier sind auch Merkmalsextraktion (z.B. TF vs. TF-IDF) und Dimensionalitätsreduktion direkt steuerbar.

Eine weitere Funktion ist die Hyperparameteroptimierung für jedes Verfahren mit anschließender Auswertung der Ergebnisse. Für die Optimierung wird eine Rastersuche mit Kreuzvalidierung verwendet.

Schließlich ist es mit dem Programm noch möglich, einen Vergleich der fünf Klassifikatoren anzustellen. Die voreingestellten Hyperparameter sind jene, welche bei der Hyperparameteroptimierung am besten abgeschnitten haben, sodass eine Vergleichbarkeit hergestellt wird.

Zur Verarbeitung und Klassifikation des Datensatzes wurden ausschließlich Klassen und Funktionen aus der „scikit-learn“-Bibliothek von Pedregosa et al. (2011) verwendet.

¹<https://github.com/rokoel02/trolldetector>

6 Evaluation

Im nachfolgenden Kapitel sollen nun die Ergebnisse der praktischen Anwendung der Klassifikationstechniken auf die vorliegenden Datensätze ausgewertet werden. Nach der Besprechung jener Optimierungsergebnisse im Einzelnen, sollen die Performances der Verfahren abschließend noch in einen Gesamtzusammenhang gestellt und verglichen werden.

6.1 Hyperparameteroptimierungen

Zu Beginn soll jedes Verfahren einzeln auf die Eignung als Trollerkennungsinstrument untersucht werden. Hierzu werden die Ergebnisse der Hyperparameteroptimierungen diskutiert. Folgende Hyperparameter teilen sich dabei alle Verfahren, da die Merkmalsextraktion bei ihnen gleich abläuft:

Hyperparameter	gewählte Werte
Merkmalgewichtung	TF, TF-IDF
Stoppwort-Filterung	keine, englisch
n-Gramm-Extraktion	1-Gramme, 1+2-Gramme

Tabelle 4: gemeinsame Hyperparameter aller Verfahren

6.1.1 k-Nearest-Neighbor-Algorithmus

Beim KNN-Algorithmus sind neben den Hyperparametern, welche mit anderen Verfahren geteilt werden, die Abstandsmetrik und der k-Wert für die zu untersuchenden Nachbarn gegeben.

Tabelle 5 zeigt die mittleren Ergebnisse der Ausprägungen aller Hyperparameter einerseits im Vergleich untereinander und im Vergleich zur durchschnittlichen und zur besten Performance. Hieraus lassen sich verschiedene Schlussfolgerungen ziehen: Insgesamt sind beim KNN-Algorithmus ausgezeichnete Zahlen zu beobachten. Im Maximum liegen nahezu alle Kennzahlen über 95%, positiv auffallend sind hier Werte der Segreganz und der Sensitivität von über 97%. Durchschnittlich werden immer Punktzahlen von über 93% erreicht. Ein Unterschied im Abschneiden unterschiedlicher Parametereinstellungen ist bei der Merkmalgewichtung zu erkennen: TF erreicht hier ein wenig bessere Punktzahlen als TF-IDF. Die Filterung von englischen Stoppwörtern bringt im Mittel keine Verbesserung hervor. Extrahiert man nur 1-Gramme, anstatt 1-Gramme und 2-Gramme, erreicht man bis zu 1% höhere Punktzahlen.

Bei den verfahrensspezifischen Hyperparametern gibt es nur wenige Schwankungen. Beim k-Wert deutet sich an, dass ein einstelliger Wert besser abschneidet als ein zweistelliger, die Verbesserungen belaufen sich aber nur auf höchstens 0,5%. Bei den Metriken ist auffällig, dass unterschiedliche Arten der Abstandsmessungen nahezu keinen Unterschied in der Qualität hervorbringen, die Abweichungen betragen hier höchstens 0,1%. Angesichts seiner durchweg hohen Punktzahlen in allen möglichen Gütemaßen ist der

Hyperparameter	Genauigkeit	Relevanz	Segreganz	Sensitivität	Spezifität	F_1
TF	0.951	0.937	0.965	0.964	0.937	0.949
TF-IDF	0.949	0.932	0.966	0.964	0.935	0.948
engl. Stoppwörter	0.947	0.931	0.962	0.961	0.934	0.946
keine Filterung	0.953	0.937	0.968	0.967	0.940	0.952
1-Gramme	0.955	0.941	0.969	0.967	0.967	0.954
1+2-Gramme	0.945	0.928	0.962	0.961	0.961	0.944
$k = 5$	0.953	0.938	0.967	0.966	0.940	0.952
$k = 15$	0.950	0.934	0.965	0.964	0.936	0.949
$k = 25$	0.948	0.931	0.964	0.962	0.934	0.947
euklid. Metr.	0.950	0.935	0.965	0.963	0.937	0.949
Manhattan	0.950	0.934	0.966	0.964	0.937	0.949
Maximum	0.960	0.948	0.973	0.972	0.950	0.959
durchschnittl.	0.950	0.934	0.965	0.964	0.937	0.949

Tabelle 5: Ergebnisse bei KNN

k-Nearest-Neighbor-Algorithmus sehr gut für die Erkennung von IRA-ähnlichen Trollen geeignet.

6.1.2 Naiver Bayes-Klassifikator

Der einzige verfahrensspezifische Hyperparameter des Naiven Bayes-Klassifikators ist die angenommene Verteilung der Merkmalvektoren. Getestet wurde mit einer Normalverteilung, einer Multinomialverteilung und einer komplementären Verteilung nach >Rennie et al. (2003)<.

Hyperparameter	Genauigkeit	Relevanz	Segreganz	Sensitivität	Spezifität	F_1
TF	0.747	0.772	0.767	0.685	0.804	0.694
TF-IDF	0.702	0.798	0.736	0.571	0.824	0.536
engl. Stoppwörter	0.715	0.751	0.731	0.620	0.803	0.616
keine Filterung	0.734	0.819	0.771	0.636	0.825	0.614
1-Gramme	0.735	0.811	0.761	0.642	0.642	0.631
1+2-Gramme	0.713	0.758	0.741	0.613	0.613	0.599
Normal	0.807	0.762	0.868	0.878	0.741	0.815
Multinomial	0.585	0.841	0.568	0.198	0.947	0.253
Komplementär	0.781	0.752	0.818	0.808	0.755	0.777
Maximum	0.834	1.000	0.949	0.958	1.000	0.855
durchschnittl.	0.724	0.785	0.751	0.628	0.814	0.615

Tabelle 6: Ergebnisse bei NB

Die Ergebnisse in Tabelle 6 lassen folgende Schlüsse zu: Dieses Verfahren schneidet je nach Parameter-Einstellung sehr unterschiedlich ab. So liefert die Annahme einer Normalverteilung oder einer komplementären Verteilung Punktzahlen von 75 - 85%. Bei einer Multinomialverteilung sind die Ergebnisse im Durchschnitt deutlich schlechter. Beispielsweise ist die mittlere Treffergenauigkeit 58,5% und die mittlere Sensitivität bei nur 19,8%. Es gibt hierbei sehr starke, gleichzeitige Ausreißer nach oben und unten: In Kombination mit TF-IDF wird eine Spezifität von 100% bei einer Sensitivität von 0% erreicht, weshalb die Treffergenauigkeit hier etwa 50% beträgt.

Bei der Merkmalgewichtung schneiden TF und TF-IDF in einer Hälfte der Gütemaße besser ab. Das Unterlassung einer Filterung von englischen Stoppwörtern führt auch bei diesem Verfahren zu leicht besseren Ergebnissen von 2 - 6%.

Lässt man die Annahme einer Multinomialverteilung außen vor, so werden mit dieser Methode Punktzahlen von durchschnittlich 75% - 85% erreicht, was gerade mit Einstellung der richtigen Hyperparameter zu relativ verlässlichen Ergebnissen führt. Unter diesen Voraussetzungen ist das Verfahren zur Trollerkennung durchaus geeignet.

6.1.3 Support Vector Machine

Die Klassifikation mit einer Support Vector Machine hat wie in Kapitel 3.3 beschrieben den Bestrafungsparameter C als einzigen verfahrensspezifischen Hyperparameter.

Hyperparameter	Genauigkeit	Relevanz	Segreganz	Sensitivität	Spezifität	F_1
TF	0.875	0.813	0.959	0.964	0.793	0.882
TF-IDF	0.868	0.838	0.900	0.900	0.837	0.868
engl. Stoppwörter	0.865	0.816	0.929	0.931	0.804	0.869
keine Filterung	0.878	0.834	0.930	0.933	0.826	0.880
1-Gramme	0.870	0.820	0.932	0.935	0.935	0.874
1+2-Gramme	0.873	0.830	0.928	0.929	0.929	0.876
$C = 1.00$	0.872	0.826	0.930	0.932	0.815	0.875
$C = 0.75$	0.871	0.825	0.929	0.932	0.815	0.875
$C = 0.50$	0.871	0.825	0.930	0.932	0.814	0.875
Maximum	0.892	0.867	0.970	0.974	0.869	0.892
durchschnittl.	0.871	0.825	0.930	0.932	0.815	0.875

Tabelle 7: Ergebnisse bei SVM

Bei Ausführung der Optimierung kommen folgende Ergebnisse (Tabelle 7) zustande: Insgesamt liefert die SVM im Mittel Punktzahlen von mindestens 80% in allen betrachteten Gütemaßen. Beste Werte sind Segreganz und Sensitivität: Durchschnittlich werden hier ca. 93% und im Maximum 97% erreicht.

Unterschiedliche Merkmalgewichtungen führen auch hier zu unterschiedlichen Ergebnissen. Sowohl TF als auch TF-IDF schneiden in drei von sechs Gütemaßen besser ab als das jeweils andere. Der stärkste Unterschied betrifft Segreganz und Sensitivität: Hier schneidet TF 6% besser ab. Bei allen anderen Arten der Merkmalsextraktion gibt es nur wenige Schwankungen von höchstens 2%.

Beim Bestrafungsparameter C sind bis auf 0,1% keine Schwankungen in den unterschiedlichen Ausprägungen auszumachen.

Insgesamt ist die SVM-Klassifikation mit den erreichten Punktzahlen ziemlich verlässlich und daher für die Trollerkennung gut geeignet. Von Vorteil ist auch, dass die durchweg hohe Segreganz für ein niedriges f_n spricht, was eine konservative Klassifikation, welche für die Trollerkennung wichtig ist, gewährleistet. Eine positive Eigenschaft ist außerdem die Stabilität des Verfahrens: Selbst mit wenigen bewussten oder zufälligen Einstellungen liefert das Verfahren immer noch gute Ergebnisse, was für eine Art Benutzerfreundlichkeit spricht.

6.1.4 Entscheidungsbäume

Bei der Entscheidungsbaum-Klassifikation sind neben den von allen geteilten Hyperparametern auch das Aufteilungskriterium gegeben. Die zwei Möglichkeiten sind hier die in Kapitel 3.4 beschriebenen Maße Gini Impurity und Entropie. Aus Tabelle 8 lassen sich folgende Schlüsse ziehen:

Die Klassifikation erreicht im Mittel in allen Gütekriterien sehr hohe Werte von 94%. Die maximalen Werte liegen mit ca. 95% nur 1% darüber. Es lässt sich aus diesem Grund leicht erkennen, dass hier nur sehr geringe Schwankungen vorhanden sind. Dies ist auch im Vergleich unter den Hyperparametern zu erkennen: Zwischen TF und TF-IDF, der Filterung englischer Stoppwörter und keiner Filterung und der Auswahl der jeweiligen n -Gramme liegen jeweils nur 1% Unterschied. Die Aufteilungskriterien Gini Impurity und Entropie unterscheiden sich nur um 0,2%.

Diese ausgezeichneten Punktzahlen legen nahe, dass eine zuverlässige Trollerkennung mit diesem Verfahren sehr gut zu leisten ist. Sowohl die geringen Schwankungen unter den Hyperparametern, als auch die geringen Unterschiede in den Gütekriterien sprechen für eine hohe Stabilität des Verfahrens. Dies bedeutet auch hier, dass selbst zufällige Einstellungen niemals zu schlechten Ergebnissen führen können.

Hyperparameter	Genauigkeit	Relevanz	Segreganz	Sensitivität	Spezifität	F_1
TF	0.945	0.938	0.951	0.948	0.941	0.943
TF-IDF	0.936	0.930	0.941	0.937	0.934	0.934
engl. Stoppwörter	0.938	0.932	0.943	0.939	0.936	0.936
keine Filterung	0.942	0.935	0.949	0.946	0.939	0.941
1-Gramme	0.944	0.937	0.951	0.948	0.948	0.943
1+2-Gramme	0.936	0.931	0.941	0.937	0.937	0.934
Gini	0.939	0.933	0.945	0.942	0.937	0.937
Entropie	0.941	0.935	0.947	0.944	0.939	0.939
Maximum	0.950	0.944	0.957	0.954	0.947	0.949
durchschnittl.	0.940	0.934	0.946	0.943	0.938	0.938

Tabelle 8: Ergebnisse der Entscheidungsbaum-Klassifikation

6.1.5 Mehrschichtiges Perzeptron

Der wichtigste einstellbare Hyperparameter bei der Klassifikation mit einem Mehrschichtigen Perzeptron ist die Aktivierungsfunktion. Bei der Hyperparameteroptimierung wurden die drei in Kapitel 3.5 beschriebenen Aktivierungsfunktionen getestet.

Hyperparameter	Genauigkeit	Relevanz	Segreganz	Sensitivität	Spezifität	F_1
TF	0.904	0.868	0.947	0.948	0.863	0.906
TF-IDF	0.888	0.868	0.909	0.907	0.870	0.887
engl. Stoppwörter	0.891	0.862	0.925	0.924	0.860	0.892
keine Filterung	0.901	0.873	0.931	0.931	0.873	0.901
1-Gramme	0.898	0.868	0.932	0.932	0.932	0.898
1+2-Gramme	0.894	0.868	0.925	0.923	0.923	0.894
relu	0.931	0.917	0.945	0.942	0.920	0.929
tanh	0.887	0.858	0.918	0.918	0.858	0.887
logistic	0.871	0.829	0.922	0.923	0.822	0.873
Maximum	0.937	0.928	0.968	0.972	0.933	0.936
durchschnittl.	0.896	0.868	0.928	0.928	0.866	0.896

Tabelle 9: Ergebnisse der MLP-Klassifikation

Tabelle 9 sind nun folgende Aussagen zu entnehmen: Im Durchschnitt liefert die MLP-Klassifikation Werte von mindestens 86%. Maximal werden Punktzahlen von 93 - 97% erreicht. Letztere Werte sind überwiegend auf die Aktivierungsfunktion ReLu zurückzuführen, welche in allen Gütemaßen 3 - 6% besser abschneidet als die anderen beiden.

Auch bei den Parametern der Merkmalsextraktion sind diesmal deutlichere Unterschiede festzustellen. Die Merkmalgewichtung mit TF liefert, analog zu allen anderen Klassifikationsverfahren, in den meisten Gütemaßen deutlich bessere Ergebnisse als TF-IDF. Filtert man englische Stoppwörter aus den Tweets, führt dies zu 1% schlechteren Ergebnissen, als wenn man dies unterlassen würde. Die ausschließliche Extraktion von 1-Grammen ist leicht besser als die Extraktion von 1- und 2-Grammen.

Die MLP-Klassifikation ist durch ihre hohen Punktzahlen in all ihren Varianten grundsätzlich für die Trollerkennung geeignet. In besonderem Maße gilt dies, wenn ReLu als Aktivierungsfunktion eingesetzt wird. Mit gut eingestellten Hyperparametern ist dieses Verfahren äußerst verlässlich. Interessant ist in diesem Zusammenhang folgendes: Die Trainingsphase mit Backpropagation wird bei einem Mehrschichtigen Perzeptron dann beendet, wenn sich der Klassifikationsscore n Iterationen in Folge nicht um einen Toleranzwert tol ändert. Um eine akzeptable Laufzeit zu gewährleisten, wurden die Werte bei der Hyperparameteroptimierung auf $n = 5$ und $tol = 0,25\%$ gesetzt. Dies bedeutet, dass sich bei Inkaufnahme einer langen Laufzeit die Punktzahlen in dem ein oder anderen Gütemaß um wenige Prozentpunkte verändern können.

6.2 Verfahren im Vergleich

Abschließend werden nun alle Ergebnisse in einen Gesamtzusammenhang eingeordnet. Tabelle 10 zeigt die Ergebnisse einer Rastersuche über alle fünf optimierten Algorithmen der Textklassifikation.

Verfahren	Genauigkeit	Relevanz	Segreganz	Sensitivität	Spezifität	F_1
KNN	0.958	0.943	0.973	0.972	0.945	0.957
NB	0.830	0.779	0.898	0.907	0.759	0.838
SVM	0.892	0.866	0.919	0.918	0.868	0.892
Baum	0.943	0.937	0.950	0.947	0.940	0.942
MLP	0.934	0.926	0.943	0.939	0.929	0.932
durchschnittl.	0.911	0.890	0.937	0.937	0.888	0.912

Tabelle 10: Ergebnisse im Vergleich

Im Durchschnitt werden mit allen Verfahren sehr gute Punktzahlen von mindestens 89% erreicht. Bei Segreganz und Sensitivität sind sogar fast 94% zu beobachten. Diese Werte sprechen dafür, dass sowohl Troll-Tweets als auch Nichttroll-Tweets mit einer hohen Zuverlässigkeit erkannt werden und dass die Gefahr einer Verwechslung eher gering ist. Somit kann für alle Verfahren eine grundsätzliche Eignung als Trollerkennungsinstrument festgestellt werden. Welches von ihnen in einem konkreten Fall zur Anwendung kommen sollte ist jedoch stets abhängig von den an die Ergebnisse gestellten Anforderungen.

Ist Genauigkeit und Zuverlässigkeit das maßgebliche Kriterium, so ist die Wahl des k-Nearest-Neighbor-Algorithmus zu empfehlen. Dieser hat mit 94 - 97% erkennbar die besten Ergebnisse aller Verfahren. Die Gefahr der Falschklassifikation ist hier mit unter 5% besonders gering.

Wird jedoch Wert auf eine gute Laufzeit gelegt, so ist der kNN-Algorithmus nicht die erste Wahl. Bei einer gegebenen Anzahl der Trainingsvektoren N und die Anzahl der Dimensionen d kann man die Zeitkomplexität mit $\Theta(dN + kN)$ angeben, sofern die Abstände gespeichert werden und mit $\Theta(kdN)$, falls dies nicht der Fall ist. Die Entscheidungsbaum-Klassifikation hat bei fast gleichwertiger Zuverlässigkeit eine Worst-Case-Laufzeit von $\mathcal{O}(dN \cdot \log(N))$ (Sani et al., 2018), schneidet jedoch im Mittel deutlich besser ab und hat hier deshalb einen entscheidenden Vorteil.

Es sind in der Anwendung auch Szenarien denkbar, in denen eine Flexibilität bezüglich des Tradeoffs Genauigkeit versus Laufzeit gewünscht wird. Für diesen Fall ist die Klassifikation mit einem Mehrschichtigen Perzeptron zu empfehlen, da die Trainingszeit hier direkt steuerbar ist. Werden schnelle Ergebnisse gewünscht und ist eine geringere Genauigkeit tolerabel so kann die Trainingszeit kurz gewählt werden. Sollen die Ergebnisse jedoch genau sein und eine höhere Laufzeit kann in Kauf genommen werden, dann kann eine längere Trainingszeit einberaumt werden.

Die Klassifikation mittels Support Vector Machine ist merkbar ungenauer als die drei hier zuerst genannten Verfahren und deshalb für die Trollerkennung eher zweite Wahl. Die Naive Bayes-Klassifikation ist als das mit Abstand ungenaueste Verfahren am wenigsten für die Anwendung geeignet.

7 Abschließendes

7.1 Fazit

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

7.2 Ausblick

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Literatur

- Kirk Baker (Jan. 2013). „Singular Value Decomposition Tutorial“. In: 2005.
- K.-L Du und M.N.s Swamy (Dez. 2014). „Multilayer Perceptrons: Architecture and Error Backpropagation“. In: S. 83–126.
- Patrick Gensing und Lena Kampf (2020). *Wie Trolle im Wahlkampf manipulierten*. URL: <https://www.tagesschau.de/faktenfinder/inland/manipulation-wahlkampf-101.html>.
- Gongde Guo, Hui Wang, David Bell und Yaxin Bi (Aug. 2004). „KNN Model-Based Approach in Classification“. In.
- Mohammad Hossin und Sulaiman M.N (März 2015). „A Review on Evaluation Metrics for Data Classification Evaluations“. In: *International Journal of Data Mining Knowledge Management Process* 5, S. 01–11.
- Philip N Howard, Bharath Ganesh, Dimitra Liotsiou, John Kelly und Camille François (2019). *The IRA, social media and political polarization in the United States, 2012-2018*, S. 3.
- I.T. Joliffe (2002). *Principal Component Analysis*. 2. Aufl. Springer Series in Statistics. Springer.
- Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes und Donald Brown (2019). „Text Classification Algorithms: A Survey“. In: *Information* 10.4.
- Philip Kreißel, Julia Ebner, Alexander Urban und Jakob Guhl (Juli 2018). *Hass auf Knopfdruck. Rechtsextreme Trollfabriken und das Ökosystem koordinierter Hasskampagnen im Netz*. URL: http://www.isdglobal.org/wp-content/uploads/2018/07/ISD_Ich_Bin_Hier_2.pdf.
- Darren L. Linvill und Patrick L. Warren (2018). *Troll Factories: The Internet Research Agency and State-Sponsored Agenda Building*.
- Christopher D. Manning und Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, S. 124–125.
- Janmenjoy Nayak, Bighnaraj Naik und Behera H. S. (2015). „A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications Challenges“. In: *International Journal of Database Theory and Application* 8.1, S. 169–186.
- Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan und Stephen Marshall (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. arXiv: [1811.03378](https://arxiv.org/abs/1811.03378) [cs.LG].
- 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 und E. Duchesnay (2011). „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12, S. 2825–2830.
- Juan Ramos (2003). *Using TF-IDF to Determine Word Relevance in Document Queries*.
- Irina Rish (Jan. 2001). „An Empirical Study of the Naïve Bayes Classifier“. In: *IJCAI 2001 Work Empir Methods Artif Intell* 3.
- Lior Rokach und Oded Maimon (Jan. 2005). „Decision Trees“. In: Bd. 6, S. 165–192.
- Habiba Muhammad Sani, Ci Lei und Daniel Neagu (2018). „Computational Complexity Analysis of Decision Tree Algorithms“. In: *Artificial Intelligence XXXV*. Hrsg. von Max Bramer und Miltos Petridis. Springer International Publishing, S. 191–197.

Qifan Zhou, Hai Zhang, Zahra Lari, Zhenbo Liu und Naser El-Sheimy (Okt. 2016). „Design and Implementation of Foot-Mounted Inertial Sensor Based Wearable Electronic Device for Game Play Application“. In: *Sensors* 16, S. 1752.

Abbildungsverzeichnis

1	Beispiel eines IRA-Tweets	1
2	Pipeline der Textklassifikation (Kowsari et al., 2019)	4
3	Tokenisierung eines Beispielsatzes	4
4	KNN-Klassifizierung für $k = 3$	7
5	SVM-Klassifizierung (Zhou et al., 2016)	9
6	Entscheidungsbaum zur Klassifizierung eines Messestandorts	10
7	Beispiel für mehrschichtiges Perzeptron	11
8	Zweifache Kreuzvalidierung	16
9	Erklärte Varianz der Komponenten	17

Tabellenverzeichnis

1	Verteilung der Hashtags	2
2	Vergleich der Datensätze	3
3	Konfusionsmatrix der Trollerkennung	13
4	gemeinsame Hyperparameter aller Verfahren	19
5	Ergebnisse bei KNN	20
6	Ergebnisse bei NB	20
7	Ergebnisse bei SVM	21
8	Ergebnisse der Entscheidungsbaum-Klassifikation	22
9	Ergebnisse der MLP-Klassifikation	23
10	Ergebnisse im Vergleich	24