Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Hussein Nour Alhuda

Erkennung von Spam-Emails mit Ansätzen des Maschinellen Lernens

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Hussein Nour Alhuda

Erkennung von Spam-Emails mit Ansätzen des Maschinellen Lernens

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Wirtschaftsinformatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Olaf Zukunft

Eingereicht am: 01. März 2022

Hussein Nour Alhuda

Thema der Arbeit

Erkennung von Spam-Emails mit Ansätzen des Maschinellen Lernens

Stichworte

Text Mining, Data Mining, Spam-E-Mail Erkennung, Text Vorverarbeitung, E-Mail Klassifikation, Spammer, Verarbeitung Natürlicher Sprache, TF-IDF, Word2Vec

Kurzzusammenfassung

E-Mails sind eine gängige und bevorzugte Methode der schriftlichen Kommunikation in unserem täglichen Leben. Das Problem mit E-Mails ist Spam. Spam-E-Mails werden mit unterschiedlichen Absichten versendet, wobei Werbung und Betrug die Hauptursachen sind. Da sie billig zu versenden sind, verursachen sie viele Probleme für die Internet-Community. Diese Arbeit befasst sich mit der Spam-E-Mails-Problematik. Dabei werden zwei unterschiedliche Merkmalsextraktionsmethoden zusammen mit sechs verschiedenen Klassifikatoren für überwachtes maschinelles Lernen untersucht. Die Leistung der durchgeführten Experimente wird anhand vier Metriken auf einem öffentlich verfügbaren Filter-Spam-Datensatz bewertet und präsentiert.

Hussein Nour Alhuda

Title of the paper

Detection of spam email using machine learning approaches

Keywords

Text Mining, Data Mining, Spam email detection, Text Preprocessing, email classification, Spammers, Natural Language Processing, TF-IDF, Word2Vec

Abstract

Email is a common and preferred method of written communication in our daily lives. The problem with email is spam. Spam emails are sent with different intentions, but advertising and scams are the main causes. Since it is cheap to ship, it causes many issues for the internet community. This thesis deals with the concern of spam e-mails. Two different feature extraction methods are examined, along with six different supervised machine learning classifiers. The performance of the experiments performed is evaluated and presented using four metrics on a publicly available filter spam dataset.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Nutzung von E-Mail ist nicht mehr auf die Kommunikation mit einer anderen Person beschränkt. In Zeiten der Digitalisierung wird ein Nutzer immer per E-Mail über aktuelle Angebote, seinen Bestellstatus oder Neuigkeiten in sozialen Netzwerken informiert.

Kürzlich wurden unerwünschte Werbe-/Massen-E-Mails, auch bekannt als Spam, zu einem großen Problem im Internet. Spam ist Verschwendung von Zeit, Speicherplatz und Kommunikationsbandbreite. Das Problem der Spam-E-Mail nimmt seit Jahren zu. Derzeit werden täglich mehr als 333 Milliarden E-Mails von 4,2 Milliarden E-Mail-Konten versendet. Bis 2023 werden voraussichtlich täglich rund 347 Milliarden E-Mails von 4,3 Milliarden Konten versendet. Dies würde eine Steigerung von ca. 20,5 % bei den täglich versendeten E-Mails und ca. 11,5 % bei neuen Konten bedeuten [22].

Knowledge Engineering und maschinelles Lernen sind die beiden allgemeinen Ansätze, die bei der E-Mail-Filterung verwendet werden. Beim Knowledge-Engineering-Ansatz muss ein Regelwerk festgelegt werden, nach dem E-Mails als Spam oder Ham kategorisiert werden. Ein Satz solcher Regeln sollte entweder vom Benutzer des Filters oder von einer anderen Behörde (z. B. dem Softwareunternehmen, das ein bestimmtes regelbasiertes Spam-Filter-Tool anbietet) erstellt werden. Die Anwendung dieser Methode zeigt keine vielversprechenden Ergebnisse, da die Regeln ständig aktualisiert und gepflegt werden müssen, was Zeitverschwendung und für die meisten Benutzer nicht bequem ist. Der Ansatz des maschinellen Lernens ist effizienter als der Ansatz des Knowledge Engineering; es erfordert keine Festlegung von Regeln [24]. Statt einer Reihe von Trainingsbeispielen handelt es sich bei diesen Beispielen um eine Reihe vorklassifizierter E-Mail-Nachrichten. Ein spezieller Algorithmus wird dann verwendet, um die Klassifizierungsregeln aus diesen E-Mail-Nachrichten zu lernen. Der Ansatz des maschinellen Lernens wurde umfassend untersucht und es gibt viele Algorithmen, die bei der E-Mail-Filterung verwendet werden können.

1.1 Ziel der Arbeit

Diese Arbeit konzentriert sich auf die Erforschung von Techniken des maschinellen Lernens, um Spam-E-Mails zu erkennen. Dazu werden verschiedene vorausgewählte Merkmalsgenerierungs- und Klassifizierungsalgorithmen kombiniert und auf einen Emails-Datensatz angewendet. Das Hauptziel besteht darin, einen Einblick in die Leistung verschiedener Kombinationsalgorithmen auf dem Datensatz zu erhalten, um bei der Entscheidung zu helfen, welche Algorithmen bzw. Kombinationen für die Spam-E-Mail-Erkennung verwendet werden sollen.

1.2 Struktur der Arbeit

Nach der Einleitung wird in Kapitel 2 in die Grundlagen der Thematik eingeführt. Hierbei geht es darum, wie natürliche Sprache mit Softwaresystemen verarbeitet wird. Zudem wird in diesem Kapitel einen detaillierten Überblick über die verschiedenen Vorverarbeitungsschritte des Texts gegeben. In Kapitel 3 werden, basierend auf theoretischem Wissen über die Verarbeitung natürlicher Sprache, Hintergrundinformationen zu den verschiedenen Merkmalsextraktionsmethoden und Klassifizierungsalgorithmen bereitgestellt, die im Rahmen dieser Arbeit verwendet werden. Das 4. Kapitel bietet einen tiefergehenden explorativen Einblick in den dieser Arbeit zugrunde liegenden Datensatz und geht auf die Hintergründe der Datenstruktur ein. Die Experiment-Pipeline und Implementierungsschritte werden in Kapital 5 beschrieben. Schließlich werden die Resultate der Klassifikation-Algorithmen anhand mehrerer ausgewählter Metriken ausgewertet, visualisiert und miteinander verglichen. In Kapital 7 werden abschließend eine Zusammenfassung und ein Gesamtfazit gezogen.

2 Grundlagen

Im Laufe der Arbeit werden einige grundlegende Begriffe verwendet, die zunächst in diesem Kapitel zur weiteren Verdeutlichung vorgestellt werden. Zu Beginn werden verschiedene Definitionen des Text Minings vorgestellt. Danach werden die Konzepte des maschinellen Lernens und der Verarbeitung natürlicher Sprache erläutert und aufgezeigt, gefolgt von einer Reihe der in dieser Arbeit verwendeten Textvorverarbeitungs- und Textverständnis-Techniken.

2.1 Text Mining

Text Mining, auch bekannt als Text Data-Mining [10] oder Knowledge Discovery aus Textdatenbanken [6], bezieht sich im Allgemeinen auf den Prozess der Extraktion interessanter und nicht trivialer Muster oder Wissen aus unstrukturierten Textdokumenten. Es kann als Erweiterung von Data-Mining oder Knowledge Discovery aus (strukturierten) Datenbanken angesehen werden [5] [29].

Text Mining ist der Prozess, qualitativ hochwertige Informationen aus Text abzuleiten. Es beinhaltet "die Entdeckung neuer, zuvor unbekannter Informationen durch den Computer durch automatisches Extrahieren von Informationen aus verschiedenen schriftlichen Ressourcen [11]. Zu den schriftlichen Ressourcen können Websites, Bücher, E-Mails, Rezensionen und Artikel gehören.

Während Data-Mining sich mit strukturierten Daten – also hoch formatierten Informationen aus Datenbanken oder ERP-Systemen – befasst, setzt sich Text Mining mit unstrukturierten Textdaten auseinander. Diese sind nicht vordefiniert oder strukturiert und stammen unter anderem aus Social Media Feeds. Beide Methoden unterscheiden sich zudem im Analyse-Ansatz: Es handelt sich weder bei Data-Mining noch bei Text Mining um eine einzelne Technologie. Stattdessen nutzen beide eine Bandbreite an Werkzeugen, um aus den verfügbaren Daten wertvolle Erkenntnisse zu generieren.

2.1.1 Data-Mining

„Data-Mining ist das semi-automatische Aufdecken von Mustern mittels Datenanalyse-Verfahren in meist sehr großen und hochdimensionalen Datenbeständen“ [19].

Data-Mining hat das Ziel neues Wissen und neue Querverbindungen aus den vorhandenen Daten zu extrahieren. Unter Wissen versteht man interessante Muster, die allgemein gültig sind [5].

Dieses Wissen sollte nicht trivial, sondern nützlich sein, man muss es also sinnvoll anwenden können. Data-Mining sollte auch weitgehend automatisch ablaufen. Das Wort „weitgehend“ wurde hier bewusst abgeschwächt, die Vorverarbeitung allerdings bedarf einer intensiven Unterstützung durch den Analysten [15].

2.2 Maschinelles Lernen

Machine Learning (ML) heißt, dass Algorithmen anhand von Daten lernen. Sie passen ihr Verhalten optimal an die Datenbasis an, um einen bestimmten Output zu kreieren. Ähnlich wie im Unternehmen, in denen Entscheidungen häufig auf Basis von Key Performance Indicator (KPI) getroffen werden, besitzt das maschinelle Lernen eigene KPIs, die sogenannten Features. Sie dienen als Input für die Ausgabe des Modells.

Abbildung 2.1: Machine Learning Model

Dabei gibt es im Groben drei verschiedene Fragestellungen, die mit ML gelöst werden. Überwachtes Lernen (Supervised Learning) mit bekannter Zielgröße, unüberwachtes Lernen (Unsupervised Learning) zum Finden von Zusammenhängen und bestärkendes Lernen (Reinforcement Learning) für die Förderung des richtigen Verhaltens [32].

2.2.1 Überwachtes Lernen

Supervised Learning: eignet sich besonders, wenn eine vorhandene Gesetzmäßigkeit gelernt und angewendet werden soll. Die Einflussgrößen und zugehörigen Zielwerte sind hier in Form von Datenpunkten repräsentiert. Beim überwachten Lernen wird das Verfahren anhand von Trainingsdaten hinsichtlich der zu erledigenden Aufgabe trainiert. Die Klassenzugehörigkeit sowohl der Trainings- als auch der Testdaten ist dabei bekannt [13].

In der Praxis ist eine der größten Herausforderungen im Bereich des Supervised Learning, ausreichend und qualitativ hochwertig annotierte Datensätze zur Verfügung zu haben. Wenig Daten stellen eine Herausforderung für den Einsatz komplexer Funktionen dar [31].

Zu den Verfahren des überwachten Lernens zählen beispielsweise die Klassifikation, darauf wird im Kapital 5 ausführlich eingegangen.

2.2.2 Unüberwachtes Lernen

Unsupervised bzw. unüberwachtes Lernen ist das zweite große Feld innerhalb des Machine Learnings. Der entscheidende Unterschied zum supervised Learning liegt darin, dass unsupervised Learning nicht auf Labels in den Datensets angewiesen ist und keinen Input von außen benötigt, um Daten zu klassifizieren. Mittels Hauptkomponenten (principal component analysis, PCA) bzw. Cluster-Analysen ermöglicht unsupervised Learning das Segmentieren, Einteilen und Bündeln von Daten anhand von algorithmisch detektierten Gemeinsamkeiten innerhalb der Daten [13].

Beim unüberwachten Lernen sind die zu entdeckenden Muster nicht bekannt, es sind weder Gruppierungen noch Klassifikationen vorgegeben. Die Lösungen, die durch entsprechende Algorithmen entwickelt werden, werden folglich nicht mit vorliegenden Lösungen abgeglichen [15]. Beispiele für das unüberwachte Lernen sind die Cluster-Analyse und die Assoziationsanalyse.

2.2.3 Verstärkendes Lernen

Reinforcement Learning ist neben supervised und unsupervised Learning der dritte große Pfeiler des Machine Learnings. Der Begriff Reinforcement bzw. verstärkendes Lernen ist aus der Psychologie bzw. dem Behaviorismus entlehnt. Dem System wird ein Ziel vorgegeben, dessen Erreichen zu einer Belohnung führt. Der Weg, um das Ziel zu erreichen, ist nicht vorgegeben.

Versinnbildlicht wird oft ein Roboter skizziert, der selbstständig aus einem Labyrinth herausfinden soll. Im initialen Schritt weiß der Roboter nichts von seiner Umgebung. Er versucht jeden möglichen Schritt und kalkuliert die damit verbundene Belohnung. Trifft er auf eine

Wand oder eine Sackgasse, wird die bisher erreichte Belohnung subtrahiert. Erreicht er das Ziel, werden die Belohnungen, die auf dem Weg zum Ziel erreicht wurden, summiert.

Gerade in Systemen mit einer Vielzahl an möglichen Handlungen, ist das Ziel, die höchstmögliche Belohnung zu erhalten, durch reines Ausprobieren nach Trial and Error nicht effektiv zu erreichen. Reinforcement-Learning-Systeme benötigen daher ein System, das zwischen Erkundung und Ausbeutung entscheidet (Exploration – Exploitation Trade Off). Das System hat die Wahl, die höchste Belohnung auf Basis des bisherigen Wissens zu wählen (Ausbeutung) oder sich für weitere Erkundung im Sinne von der Suche nach neuen Wegen zu entscheiden [31].

Neben diesen drei Kategorien verwenden viele gängige Algorithmen semi-überwachtes Lernen, das, wie der Name schon sagt, eine Mischung aus überwachtem und unüberwachtem Lernen ist. Hier werden sowohl beschriftete als auch unbeschriftete Instanzen verwendet, um ein ML-Modell zu trainieren. Diese Arbeit verwendet jedoch überwachte Lernalgorithmen, da Klassenbezeichnungen für Tweets bereits gegeben sind.

2.3 Verarbeitung natürlicher Sprache (NLP)

Die Abkürzung NLP steht für Natural Language Processing und beschreibt Techniken und Methoden zur maschinellen Verarbeitung natürlicher Sprache. Ziel ist eine direkte Kommunikation zwischen Mensch und Computer auf Basis der natürlichen Sprache.

Natural Language Processing (NLP) versucht, natürliche Sprache zu erfassen und mithilfe von Regeln und Algorithmen computerbasiert zu verarbeiten. NLP verwendet hierfür verschiedene Methoden und Ergebnisse aus den Sprachwissenschaften und kombiniert sie mit moderner Informatik und künstlicher Intelligenz. Ziel ist es, eine möglichst weitreichende Kommunikation zwischen Mensch und Computer per Sprache zu schaffen. Dadurch sollen sich sowohl Maschinen als auch Anwendungen per Sprache steuern und bedienen lassen [14].

In anderen Worten Natural Language Processing ist der Prozess des Analysierens von Text, des Erstellens von Beziehungen zwischen Wörtern, des Verstehens der Bedeutung dieser Wörter und des Ableitens eines besseren Verständnisses der Bedeutung der Wörter, um daraus Informationen, Wissen oder neuen Text zu generieren

2.3.1 Anwendungsbereiche

Für folgende Anwendungsbereiche kann Natural Language Processing eingesetzt werden:

1. Spracherkennung (text to speech & speech to text).

2. Segmentierung zuvor erfasster Sprache in einzelne Wörter, Sätze und Phrasen.
3. Erkennen der Grundformen der Wörter und Erfassung grammatischer Informationen
4. Erkennen der Funktionen einzelner Wörter im Satz (Subjekt, Verb, Objekt, Artikel, etc.)
5. Extraktion der Bedeutung von Sätzen und Satzteilen bzw. Phrasen wie Adjektiv-Phrasen (z.B. zu langen), Präpositionalphrasen (z.B. an den Fluss) oder Nominalphrasen (z.B. der zu langen Party)
6. Erkennen von Satzzusammenhängen, Satzbeziehungen und Entitäten.

Natural Language Processing kann sowohl für die linguistische Textanalyse, Stimmungs- und Meinungs-Analyse (Sentiment-Analyse), Übersetzungen, als auch für Sprachassistenten, Chatbot und zugrunde liegenden Frage- und Antwort-Systemen zum Einsatz kommen.

Im Kombination mit den Aspekten der Informatik, wie Systemanalyse und Modellierung und Bereichen der theoretischen Informatik, beispielsweise Berechenbarkeit und Komplexitätstheorien, wird der Computerlinguistik Wissen über Datenstrukturen und der Verwendung effizienter Verfahren zugesteuert. Somit können praktische Anwendungsfälle bearbeitet werden, wie die Übersetzung eines Satzes in verschiedenen Sprachen oder die Bearbeitung einer telefonischen Pizzabestellung. Um diese Ziele zu erreichen, sind diverse Aufgabenfelder zu besetzen. Sie erreichen von der Erkennung gesprochener Sprachen, über den Bau eines Parsers bis hin zur Ansammlung von Methoden und Algorithmen, mit denen die natürliche Sprache/ Kommunikation des Menschen in einem Softwaresystem analysiert werden kann [3].

Dabei ist es wichtig, ein möglichst weitreichendes Verständnis von gesprochener oder geschriebener Sprache zu extrahieren. Voraussetzung dafür ist eine Quelle an textuellen Daten, auch Korpus genannt. Da Korpusse und andere textuelle Daten in ihrem Querformat normalerweise unstrukturiert in Erscheinung treten, wird eine Bandbreite an Techniken angewendet, und Rohdaten in gut-definierte Sequenzen aus linguistischen Komponenten einzugliedern [26].

Diese intensive Vorverarbeitung ist gerade deshalb nötig, da Texte häufig mit einer Vielzahl an unterschiedlichen Daten, wie HTML-Tags, Hyperlinks und andere irrelevante Einheiten auftreten. Ferner haben unterschiedliche Wörter unterschiedliche Wichtigkeiten. Beispielsweise haben häufig vorkommende Wörter wie Artikel(„der“, „die“, „das“) oder Konjunktionen (zum Beispiel „und“, „oder“, „weil“) im Umfeld des Text-Minings eine sehr geringe Wichtigkeit.

2.3.2 Herausforderung

Natural Language Processing befasst sich damit, die menschliche Sprache zu verstehen – und genau dort liegt das große Problem. Die Sprache ist nämlich sehr komplex und nicht immer eindeutig. Zusätzlich wird das Auswerten von menschlicher Sprache durch beispielsweise Redewendungen, Ironie oder Wortspiele zusätzlich enorm erschwert. Es reicht nicht, dass der Computer einzelne Wörter erkennt. Entscheidend sind die Sätze und Zusammenhänge, die die Maschine verstehen muss. Der Mensch hat hierbei den Vorteil, dass er auf seine Lebenserfahrung zurückblicken kann und die Sprache quasi von Geburt an lernt. Daher ist Sprache für den Menschen leicht verständlich und einzuordnen. Der Computer hingegen muss mithilfe von Algorithmen diesen Status erst erlernen. Die Regeln, nach denen die Sprache ausgewertet wird, können unterschiedlich komplex sein. Beispielsweise ist es relativ einfach durch Wortendungen zu erkennen, ob es sich um Singular oder Plural eines Begriffs handelt. Schwierig wird es hingegen, wenn Sarkasmus verwendet wird, da dieser oftmals nur durch Betonung deutlich wird.

2.4 Text Vorverarbeitung

Der Text-Mining-Prozess umfasst die Textvorverarbeitung, die Merkmal- und Auswahlgenerierung, die Musterextraktion bis hin zur Analyse der Ergebnisse [28].

Es ist notwendig, die Daten vor der Anwendung der Data-Mining-Verfahren aufzubereiten, damit die entwickelten Modelle qualitativ hochwertige und aussagekräftige Ergebnisse erzielen. Es werden fehlende Werte und Inkonsistenzen innerhalb der Daten behoben, sowie Transformationen der Daten durchgeführt.

Alle maschinellen Lernalgorithmen, seien es überwachte oder unüberwachte Techniken, arbeiten mit Eingabefunktionen, die numerischer Natur sind, um zu diesem Schritt zu gelangen, muss man jedoch die anfänglichen Textdaten bereinigen, normalisieren und vorverarbeiten.

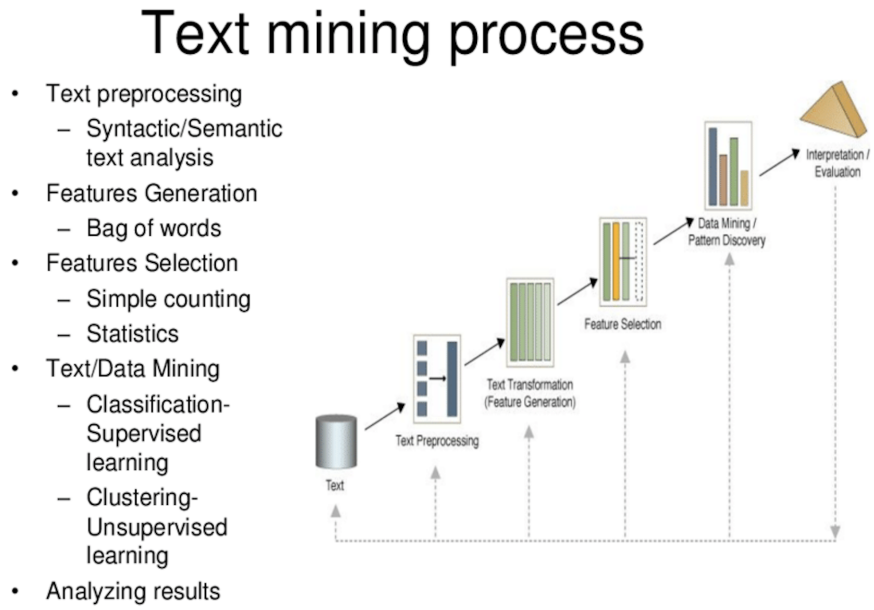


Abbildung 2.2: Text Mining Process [28]

Die Textkorpora und andere Textdaten in ihren nativen Rohformaten sind normalerweise nicht gut formatiert und standardisiert und das ist natürlich zu erwarten, schließlich sind Textdaten sehr unstrukturiert. Textverarbeitung oder, genauer gesagt, Vorverarbeitung, umfasst eine Vielzahl von Techniken, die Rohtext in wohldefinierte Sequenzen linguistischer Komponenten mit Standardstruktur und -notation umwandeln. Zusätzliche Metadaten sind oft auch in Form von Annotationen vorhanden. Die folgende Liste verfasst eine Vorstellung von einigen der beliebtesten Textvorverarbeitungs- und Textverständnis-Techniken [26].

- Grundlegende Textbereinigung.
- Tokenisierung
- Stemming
- Lemmatisierung
- Entfernen von Stoppwörter

2.4.1 Textbereinigung

Die Extraktion von verwertbaren Daten ist in diesem Fall von höchster Wichtigkeit, da eine robuste Bereinigung des Texts, sei es in Worten, Sätzen oder Absätzen, eine massive Auswirkung

auf die restlichen, höher komplexen Verarbeitungsschritte der Anwendung haben kann. Ein bereinigtes Dokument sorgt somit dafür, dass ungewollte Ausgaben und verfälschte Ergebnisse durch verrauschte Daten vermieden werden können [26].

Als Rauschfaktoren können Auszeichnungssprachen oder andere Formatierungs- und Gestaltungsmethoden elektronischen Ursprungs in Verbindung gebracht werden. Beispiele sind HTML-Tags oder Inline-CSS-Strukturen. Das Ignorieren dieser Einheiten ist ein wesentlicher Schritt beim Bereinigen des Textes. Darüber hinaus, das Entfernen von Zahlen, URLs, Hashtags, Symbolen, Emojis oder Zeichentrennzeichen kann die Textreinheit erheblich verbessern [30].

2.4.2 Tokenisierung

Ein weiterer Verarbeitungsschritt des Natural Language Processing stellt die Tokenisierung dar. Die Tokenisierung beschäftigt sich mit der Aufteilung von Sätzen oder Texten in einzelnen Token anhand von Leerzeichen, Punktation, oder Sonderzeichen mit dem Ziel, den textuellen Eingangswert für weitere Verarbeitungsschritte sinnvoll in linguistische Einheiten zu listen. Je nach Verfahren und Anwendungsfälle können sich Werte, die in einem Token aufgenommen werden, unterscheiden [3]. Im Folgenden werden drei tokenbasierte Verfahren und deren Funktionsweise anhand eines Beispiels erläutert, um die Funktionsweise von Tokenisierung zu verdeutlichen.

Die Leerzeichen-Tokenisierung ist die einfachste Form der Separierung. Hierbei werden Token mit Hilfe der vorkommenden Leerzeichen voneinander getrennt.

Abbildung 2.3: Eigene Darstellung zur Leerzeichen-Tokenisierung

Die Wort-Punkt-Tokenisierung erstellt eine Sequenz aus alphabetischen und nicht alphabetischen Zeichen:

Abbildung 2.4: Eigene Darstellung zur Wort-Punkt-Tokenisierung

Das Verfahren Treebank-Tokenisierung hingegen teilt Punktationen nur, wenn darauf ein Leerzeichen folgt. Daher findet dieses Verfahren häufig Verwendung bei englischsprachigen Anwendungsfällen, da kurz Formen wie Ï'mïn Ïünd “m"getrennt und somit die Semantik besser identifiziert werden kann [26]. Für die Tokenisierung des Beispiels ergibt sich mit der Tree-bank-Tokenisierung folgende Anordnung:

Abbildung 2.5: Eigene Darstellung zur Treebank-Tokenisierung

2.4.3 Stemming

Der Begriff Stemming steht für ein Verfahren aus der Sprachwissenschaft sowie der Informatik und bedeutet auf Deutsch so viel wie Stammformreduktion. Dabei werden Wörter auf einen gemeinsamen Wortstamm zurückgeführt

Stemmeng-Verfahren nehmen meist eine lexikonfreie Suffix-Analyse an und sind einfach an viele Sprachen anzupassen. Man beachte, dass Stemming quasi eine Komprimierung mit Verlust darstellt, da ja durch das Stemming Wörter mit unterschiedlichen Suffixen auf denselben Teilstring reduziert werden können. [3].

	original_word	stemmed_words
0	connect	connect
1	connected	connect
2	connection	connect
3	connections	connect
4	connects	connect

Abbildung 2.6: Stemming, Reduzierung auf den Wortstamm "Connect"[7]

Bei obiger Abbildung kann man die Reduzierung auf den Teilstring „Connect“ mit den unterschiedlichen Suffixen „ed“, „Ion“, „Ions“ und „s“ als Exempel der Funktionalität von Stemming-Verfahren nachvollziehen.

Da auch die Schlüsselwörter aus der Anfrage entsprechend normalisiert werden müssen, verursacht Stemming im Allgemeinen eine erhöhte Ungenauigkeit in der Volltextsuche. Dies ist auch bekannt als Overstemming und Understemming.

Overstemming ist der Prozess, bei dem ein viel größerer Teil eines Wortes abgeschnitten wird, als erforderlich ist, was wiederum dazu führt, dass zwei oder mehr Wörter auf denselben Wortstamm oder Stamm falsch reduziert werden, obwohl sie auf zwei Stammwörter oder mehr reduziert werden sollten (semantisch unterschiedliche Wortformen werden auf denselben Stamm abgebildet, z. B. ‚university‘ und ‚universe‘ auf ‚univers‘ [3]).

Bei der Understemming könnten zwei oder mehr Wörter fälschlicherweise auf mehr als ein Wurzelwort reduziert werden, obwohl sie eigentlich auf dasselbe Wurzelwort reduziert werden sollten (unterschiedliche Wortformen eines Wortes werden auf unterschiedliche Stämme abgebildet, z. B. ‚data‘ und ‚datum‘ auf ‚dat‘ und ‚datu‘ anstelle des gleichen Stammes ‚dat‘ [3]).

2.4.4 Lemmatisierung

Der Prozess der Lemmatisierung ist dem Stemming sehr ähnlich, bei dem man Wortzusätze entfernt, um zu einer Grundform des Wortes zu gelangen. Allerdings ist in diesem Fall auch diese Grundform bekannt als Wurzelwort, aber nicht als Wurzelstamm. Der Unterschied zwischen den beiden besteht darin, dass der Wurzelstamm möglicherweise nicht immer ein lexikografisch korrektes Wort ist, d. h. er ist möglicherweise nicht im Wörterbuch vorhanden, aber das Wurzelwort, auch Lemma genannt, wird immer im Wörterbuch vorhanden sein.

Der Lemmatisierungsprozess ist erheblich langsamer als das Stammen, da ein zusätzlicher Schritt erforderlich ist, bei dem die Wurzelform oder das Lemma gebildet wird, indem das Affix aus dem Wort entfernt wird, wenn und nur wenn das Lemma im Wörterbuch vorhanden ist [26]. Das erfordert zusätzliche Leistung der Computerlinguistik, wie z. B. einen Part-of-Speech-Tagger. Dadurch sind bessere Auflösungen möglich. (z. B. ‚is‘ und ‚are‘ auf ‚be‘)

2.4.5 Entfernung von Stoppwörter

Stoppwörter sind Wörter mit geringer oder keiner Bedeutung und werden normalerweise bei der Verarbeitung aus dem Text entfernt, um Wörter mit maximaler Bedeutung und Kontext zu erhalten. Stoppwörter treten normalerweise am häufigsten auf, wenn einen Textkorpus basierend auf singulären Token aggregiert und deren Häufigkeit überprüft werden. Wörter wie

„a“, „the“, „und“ usw. sind Stoppwörter. Es gibt keine universelle oder erschöpfende Liste von Stoppwörtern und oft hat jede Domäne oder Sprache ihren eigenen Satz von Stoppwörtern [26]. Um dieses Prinzip etwas deutlicher darzustellen, wird die folgende Tabelle betrachtet.

Beispieltext mit Stoppwörter	Beispieltext ohne Stoppwörter
Hier we have a computer science Portal for Geeks	Computer Science, Portal, Geeks
Can listening be so exhausting?	Listening, Exhausting
I like reading so much, so I read	Like, Reading, read

Abbildung 2.7: Eigene Darstellung zur Entfernung von Stoppwörter

2.4.6 Normalisierung

Ein stark übersehener Vorverarbeitungsschritt ist die Textnormalisierung. Textnormalisierung ist der Prozess der Umwandlung eines Textes in eine kanonische (Standard-)Form. Zum Beispiel können die Wörter „gooood“ und „gud“ auf „good“, seine kanonische Form, umgewandelt werden. Ein weiteres Beispiel ist die Zuordnung nahezu identischer Wörter wie „Stopwords“, „Stop -words“ und „Stop words“ auf „Stopwords“ [27].

Die Textnormalisierung ist wichtig für verrauschte Texte wie Social-Media-Kommentare, Textnachrichten und Kommentare zu Blog-Posts, bei denen Abkürzungen, Rechtschreibfehler und die Verwendung von Wörtern außerhalb des Vokabulars weit verbreitet sind [27].

3 Methoden

Das vorliegende Kapitel stellt die verschiedenen Repräsentationsmethoden und Algorithmen vor, die in dieser Arbeit weiter verwendet werden, indem es einen detaillierten Einblick in ihre Arbeitsweise gibt.

3.1 Features extraction

Da alle Machine-Learning-Modelle nur numerische Darstellungen von Features als Eingaben verstehen, sind sie eingeschränkt, Textdaten direkt verarbeiten zu können.

Textdaten bestehen normalerweise aus Dokumenten, die Wörter und Sätze von frei fließendem Text darstellen können. Der inhärente Mangel an Struktur (keine sauber formatierten Datenspalten!) und die verrauschte Natur von Textdaten erschweren es Methoden des maschinellen Lernens, direkt mit Rohtextdaten zu arbeiten. Daher wird in diesem Kapitel praktische Ansätze gezeigt, um einige der beliebtesten und effektivsten Strategien zum Extrahieren aussagekräftiger Merkmale aus Textdaten zu erkunden. Diese Funktionen können dann verwendet werden, um Text effizient darzustellen, was weiter genutzt werden kann, um Machine-Learning-Modelle einfach zu erstellen, um komplexe Aufgaben zu lösen [26].

3.1.1 Bag of Words

Dies ist das bekannste Vektorraum-Repräsentationsmodell für unstrukturierten Text. Ein Vektorraummodell ist ein mathematisches Modell, um unstrukturierten Text (oder beliebige andere Daten) als numerische Vektoren darzustellen, so dass jede Dimension des Vektors ein spezifisches Merkmal/Attribut ist. Der Name des Modells ist so, weil jedes Dokument buchstäblich als eine Tüte mit seinen eigenen Wörtern dargestellt wird, ohne die Wortreihenfolge, Sequenzen und Grammatik zu berücksichtigen.

Das Bag-of-Words-Modell stellt jedes Textdokument als numerischen Vektor dar, wobei jede Dimension ein bestimmtes Wort aus dem Korpus ist und der Wert seine Häufigkeit im Dokument, das Vorkommen (bezeichnet mit 1 oder 0) oder sogar gewichtete Werte sein kann.

Die Merkmalsmatrix wird traditionell als eine dünn besetzte Matrix dargestellt, da die Anzahl der Merkmale mit jedem Dokument phänomenal ansteigt, wenn man bedenkt, dass jedes einzelne Wort zu einem Merkmal wird. Man betrachtet das Paar (x, y), hier steht x für ein Dokument und y für ein bestimmtes Wort/einbestimmtes Merkmal und der Wert davon ist die Häufigkeit, mit der y in x vorkommt [26].

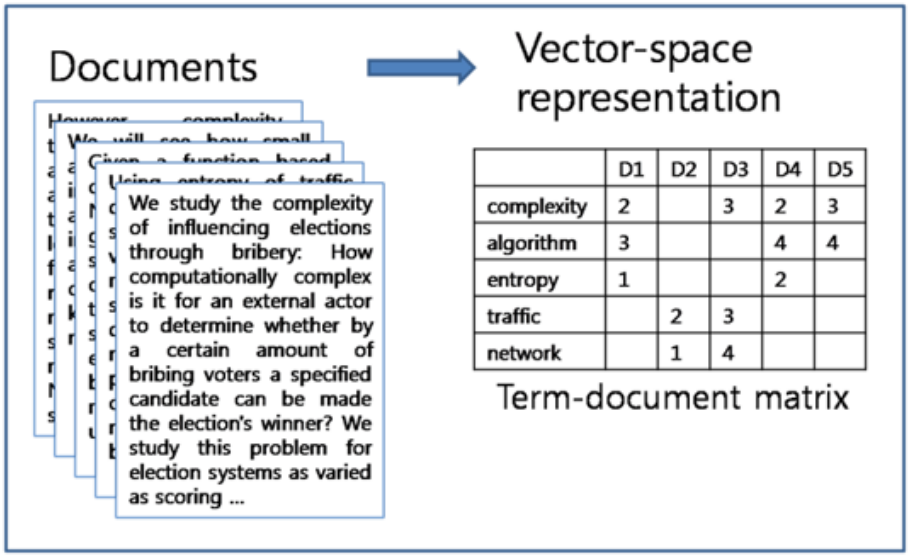


Abbildung 3.1: Term-document Matrix

Die Abbildung 6.1 zeigt, dass je mehr Dokumente man hat, desto größer das Vokabular ist, sodass die Feature-Matrix eine riesige Matrix sein wird. Daher geht dem Bag-of-Words-Modell in der Regel eine wichtige Vorverarbeitung (Wortreinigung, Entfernung von Stoppwörtern, Stemming/Lemmatisierung usw.) voraus, die darauf abzielt, das Dimensionsproblem zu reduzieren.

3.1.2 TF-IDF

Die Häufigkeit von Begriffen ist nicht unbedingt die beste Darstellung für Text. Tatsächlich findet man im Korpus gemeinsame Wörter mit der höchsten Frequenz, aber wenig Vorhersagekraft über die Zielvariable. Um dieses Problem zu beheben, gibt es eine erweiterte Variante der Bag-of-Words. TF-IDF ist eine sehr beliebte und aktuelle Strategie im Bereich Text-Mining, die den Extraktionsprozess erheblich vereinfacht. Diese Methode extrahiert die aussagekräftigsten Begriffe in einem Dokument, was häufig zu signifikanten Ergebnissen führt. Allerdings wird

beim Extrahieren von Wörtern mit TF-IDF jedes Wort im Text als separate Einheit behandelt, sodass die semantischen Merkmale des Textes nicht effektiv erhalten werden können.

TF-IDF steht für den Begriff Frequency-Inverse Document Frequency. Es ist eine Kombination aus zwei Metriken, der Begriffshäufigkeit $|tf|$ und der inversen Dokumentenhäufigkeit $|idf|$. Diese Technik wurde ursprünglich als Metrik für das Ranking von Suchmaschinenergebnissen basierend auf Benutzeranfragen entwickelt und ist mittlerweile ein Teil des Informationsabrufs und der Extraktion von Textmerkmalen [21].

Mathematisch ist TD-IDF das Produkt zweier Metriken und kann wie folgt dargestellt werden:

$$tfidf = tf \times idf \tag{3.1}$$

Die Termhäufigkeit $|tf|$ in jedem Dokumentvektor wird durch den Rohhäufigkeitswert dieses Terms in einem bestimmten Dokument bezeichnet. Mathematisch lässt es sich wie folgt darstellen:

$$tf(w, D) = f_{w,d} \tag{3.2}$$

wobei $|f_{w,d}|$ die Häufigkeit für das Wort $|w|$ in Dokument $|D|$ bezeichnet, was zum Begriff Häufigkeit $|tf|$ wird.

Die mit $|idf|$ bezeichnete inverse Dokumentenhäufigkeit ist der Kehrwert der Dokumentenhäufigkeit für jeden Begriff und wird berechnet, indem die Gesamtzahl der Dokumente im Korpus durch die Dokumentenhäufigkeit geteilt, in denen ein Begriff erscheint und dann das Ergebnis logarithmisch skaliert wird.

$$idf_t = \log \frac{|D|}{|D|_t} \tag{3.3}$$

wobei $|D|$ die Gesamtzahl der Dokumente ist, und $|D|_t$ die Anzahl der Dokumente ist, in denen der Begriff t erscheint.
Somit kann der Begriff frequenzinverse Dokumentenhäufigkeit berechnet werden, indem diese beiden Maße multipliziert werden [26].

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \tag{3.4}$$

Obwohl es sich um effektive Methoden zum Extrahieren von Merkmalen aus Text handelt, ist das Modell nur eine Tüte unstrukturierter Wörter, dabei verliert man zusätzliche Informationen wie Semantik, Struktur, Sequenz und Kontext zu benachbarten Wörter in jedem Textdokument. Es gibt fortgeschrittenere Modelle, die sich um diese Aspekte kümmern, die in einem nachfolgenden Abschnitt dieses Kapitels behandelt werden.

3.1.3 Worteinbettung

Im Allgemeinen werden Worteinbettungsansätze in der Verarbeitung natürlicher Sprache verwendet, um die Bag-of-Words-Darstellung in eine kontinuierliche räumliche Darstellung umzuwandeln [20]. Dieser kontinuierliche Raum hat einige Vorteile, da die Dimensionalität stark reduziert wird und die Wörter mit engerer Bedeutung in diesem neuen kontinuierlichen Raum eng beieinander liegen. Mehrere Anwendungen der Worteinbettung basierend auf neuronalen Netzen wurden eingeführt, darunter Word2vec, Glove und Wikitionary [16].

3.1.4 Word2Vec

Word2Vec ist eine der beliebtesten vortrainierten Worteinbettungen, die von Google entwickelt wurde. Word2Vec wird auf dem Google News-Datensatz (ca. 100 Milliarden Wörter) trainiert. Es hat mehrere Anwendungsfälle wie Recommendation Engines, Knowledge Discovery und wird auch in den verschiedenen Textklassifizierungsproblemen angewendet.

Das Word2Vec-Modell basiert auf der Annahme, dass Wörter mit ähnlicher Semantik im gleichen Kontext vorkommen (was Bag-of-Words-Modell nicht leisten kann). Das Modell verwendet einen riesigen Textkorpus als Eingabe. Es erzeugt dann einen Vektorraum, der normalerweise Hunderte von Dimensionen hat. Jedem charakteristischen Wort im Korpus ist ein entsprechender Vektor im Raum zugeordnet. Die Wörter mit gemeinsamen Kontexten werden im Vektorraum nahe beieinander platziert.

Es gibt zwei verschiedene Modellarchitekturen, die von Word2Vec genutzt werden können, um diese Worteinbettungsdarstellungen zu erstellen. Diese beinhalten:

- Das Continuous-Bag-of-Words-Modell (CBOW)
- Das Skip-Gram-Modell

Es wurden von Mikolov et al. eingeführt [17].

Continuous Bag of Words -Modell (CBOW)

Die Architektur des CBOW-Modells versucht, das aktuelle Zielwort (das mittlere Wort) basierend auf den Quellkontextwörtern (umgebenden Wörtern) vorherzusagen. Betrachtet man einen einfachen Satz, "the quick brown fox jumps over the lazy dog", können die Paare von (Kontext-Fenster, ZielWort) sein, wobei, wenn man ein Kontext-Fenster der Größe 2 betrachtet, beispielsweise wie ([quick , Fox], Brown), ([the, brown], quick), ([the, dog], lazy). Somit versucht das Modell, das Zielwort basierend auf den Kontext-Fenster-Wörtern vorherzusagen [26].

Skip-Gram-Modell

Das Skipgram-Modell ähnelt dem CBoW-Modell, aber anstatt das zentrale Wort anhand des Kontexts vorherzusagen, sagt Skipgram den Kontext anhand des zentralen Worts voraus. Dadurch kann das Skipgram-Modell viel mehr Trainingsdaten generieren, was es besser für kleine Datensätze geeignet macht. es ist jedoch auch langsamer als CBoW [18]

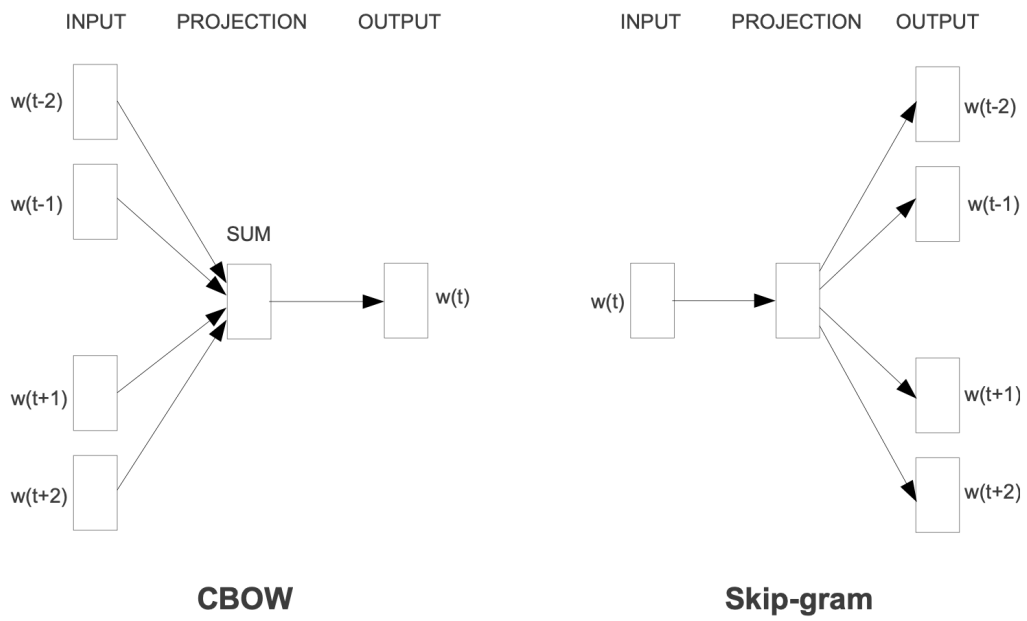


Abbildung 3.2: Neue Modellarchitekturen. Die CBOW-Architektur sagt das aktuelle Wort basierend auf dem Kontext voraus, und das Skip-gram sagt umgebende Wörter anhand des aktuellen Wortes voraus [17]

3.2 Klassifikationsverfahren

Die Klassifizierung von Spam-E-Mails wird als binäres Klassifizierungsproblem betrachtet, bei dem E-Mails als Spam oder Ham klassifiziert werden. Die beliebtesten maschinellen Lernalgorithmen bei der Spam-Klassifizierung sind Naive Bayes, Entscheidungsbaum, Support Vector Machine und Neural Network.

3.2.1 Naive Bayes

Einer der populärsten Klassifikatoren, die bei Spam-Klassifizierungsproblemen verwendet werden, ist Naive Bayes und wird auch allgemein bei Textkategorisierungsproblemen verwendet [33]. Der Naive Bayes-Klassifikator wurde ursprünglich von [25] für Probleme bei der Spam-Klassifizierung vorgeschlagen.

Die Arbeit von Sahami aus dem Jahr 1998 [25] führte zur Implementierung mehrerer Spam-Filter unter Verwendung verschiedener Algorithmen des maschinellen Lernens in Kombination mit den Empfehlungen von Graham aus dem Jahr 2002 [8] unter Verwendung der Bayesschen Analyse.

Der Bayessche Klassifikator arbeitet mit den abhängigen Ereignissen und der Wahrscheinlichkeit, dass ein Ereignis in der Zukunft eintritt, das aus dem vorherigen Auftreten desselben Ereignisses erkannt werden kann. Diese Technik kann verwendet werden, um Spam-E-Mails zu klassifizieren; Wortwahrscheinlichkeiten spielen hier die Hauptregel. Wenn einige Wörter häufig im Spam vorkommen, aber nicht im Ham, dann handelt es sich bei dieser eingehenden E-Mail wahrscheinlich um Spam. Die naive Bayes-Klassifikatortechnik ist zu einer sehr beliebten Methode in Mail-Filtersoftware geworden. Bayes-Filter sollten trainiert werden, um effektiv zu arbeiten. Jedes Wort hat eine bestimmte Wahrscheinlichkeit, in seiner Datenbank in Spam- oder Ham-E-Mails vorzukommen. Wenn die Summe der Wortwahrscheinlichkeiten einen bestimmten Grenzwert überschreitet, markiert der Filter die E-Mail in einer der beiden Kategorien. Hier sind nur zwei Kategorien notwendig Spam oder Ham.

Sei x = (x1, ... , xd) ein Datenobjekt und c in C die Klassenzugehörigkeit, dann wird die Klasse c gesucht, für die die bedingte Wahrscheinlichkeit P(c|x) am größten ist. Die bedingte Wahrscheinlichkeit wird mit Hilfe des Satzes von Bayes berechnet [19].

Satz von Bayes

$$P(c|x) = \frac{P(x|c) \times P(c)}{P(x)} \tag{3.5}$$

P(c|x) ist die Wahrscheinlichkeit von x = (x1, ... , xd) unter der Bedingung, dass x der Klasse c angehört. p(x) repräsentiert die Wahrscheinlichkeit von x. P(x) ist für alle Klassen identisch, weshalb dieser Term ignoriert werden kann. Das Ziel ist es demzufolge, die Klasse c zu finden, für die der Ausdruck P(x|c) * P(c) maximiert wird [19].

$$C^* = \arg\max P(x1, ...xd|c) \cdot P(c) \tag{3.6}$$

Formel von Naive Bayes Spam Filter

Naive Bayes Spam-Filterung ist eine Basistechnik für den Umgang mit Spam, die sich an die E-Mail-Anforderungen einzelner Benutzer anpassen und niedrige Falsch-Positiv-Spam-Erkennungsraten liefern kann, die für Benutzer im Allgemeinen akzeptabel sind.

$$P(S|W) = \frac{P(W|S) \cdot P(S)}{P(W|S) \cdot P(S) + P(W|H) \cdot P(H)} \tag{3.7}$$

- P(S|W) $\rightarrow$ Wahrscheinlichkeit, dass Die Mail Spam ist wenn Wort W in ihr vorkommt.
- P(W|S) $\rightarrow$ Wahrscheinlichkeit, dass Wort W in einer Spam Nachricht vorkommt.
- P(W|H) $\rightarrow$ Wahrscheinlichkeit, dass Wort W in einer Ham Nachricht vorkommt.
- P(S) $\rightarrow$ Wahrscheinlichkeit, dass Nachricht Spam ist.
- P(H) $\rightarrow$ Wahrscheinlichkeit, dass Nachricht Spam ist.

Die Vorteile des naiven Bayes Verfahrens sind eine hohe Klassifikationsgeschwindigkeit neuer Datenobjekte, sowie eine relativ hohe Lerngeschwindigkeit des Klassifikationsmodells auch mit vielen Trainingsdaten und Attributen. Ein weiterer Vorteil ist, dass das Klassifikationsmodell auch inkrementell aktualisiert werden kann, d. h. bei neuen Daten das Modell nicht vollständig von neuem erstellt werden muss. Ein Nachteil des naiven Bayes Verfahrens ist es, dass die Klassifikation die wechselseitige Abhängigkeit nicht nur paarweise, sondern auch höherer Ordnung von Attributwerten unberücksichtigt lässt. In einigen Anwendungen ist gerade das gemeinsame Auftreten von Attributwerten problemtypisch (etwa „online pharmacy“ bei der Spam-Erkennung), wobei jedoch das einzelne Auftreten (nur „online“ oder „pharmacy“) nicht auf eine Klasse hindeutet.

3.2.2 Entscheidungsbaum

Ein Entscheidungsbaum ist ein gerichteter Baum und besteht aus einer Menge von Knoten und Kanten. Die Knotenmenge besteht aus einem Wurzelknoten, Blattknoten mit einzelnen Klassenzuweisungen und inneren Zwischenknoten, die mit Splitting-Attributen annotiert sind. Die Wurzel- und Zwischenknoten stellen boolesche Variablen dar, die Blattknoten die Klassenzugehörigkeit eines Objekts. Die Kanten verbinden Knotenpaare miteinander und sind mit den Werten der Splitting-Attribute gekennzeichnet. Jeder Pfad vom Wurzelknoten zu einem Blattknoten repräsentiert eine Klassifizierungsregel.[19]

Ein Beispiel für einen Entscheidungsbaum zur Ermittlung eines Zahlungsausfalls ist in Abbildung3.3 dargestellt. Dort wird eine Klassifikation anhand von Kriterien wie beispielsweise Beamtenstatus und Einkommen vorgenommen.

Der Entscheidungsbaum wird anhand der Trainingsdaten konstruiert, deren Klassenzugehörigkeit bereits bekannt ist. Die Konstruktion findet rekursiv ausgehend vom Wurzelknoten statt. Es werden für jeden Knoten Attributwerte gesucht, die die Objekte in möglichst homogene Partitionen aufteilen, sodass der Klassifikationsfehler gering ist.

Der rekursive Algorithmus endet, falls keine weiteren Attribute mehr vorliegen oder die Klassenzugehörigkeit der Objekte eindeutig festgelegt wurde. Die Auswahl der geeigneten Attribute für den jeweils nächsten Split hängt von der Homogenität der erzeugten Untermengen, also der Gleichartigkeit der in den Untermengen enthaltenen Objekte ab.

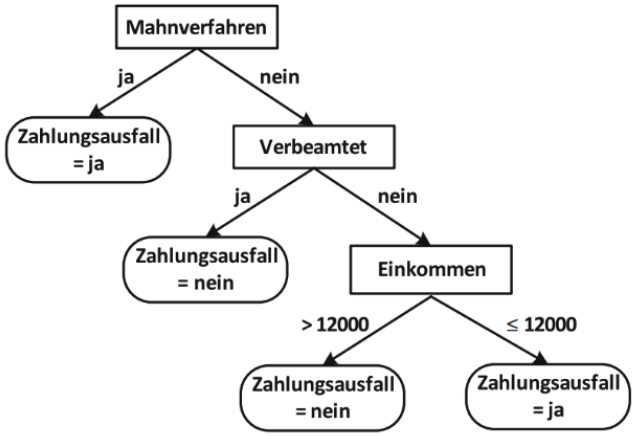


Abbildung 3.3: Entscheidungsbaum zur Ermittlung eines Zahlungsausfalls [19]

Die zwei am weitesten verbreiteten Verfahren für das Quantifizieren der Inhomogenität sind die Entropie und der Gini-Index [19].

Entropie

Die Entropie ist ein Maß für den Informationsgehalt für die Konzentration einer Objektmenge. Die Entropie einer Partition P mit k Klassen ist definiert als

$$Entropie(P) = -\sum_{i=1}^{K}(pi \times \log_2 pi) \tag{3.8}$$

mit pi als relative Häufigkeit der Klasse i in der Partition P.

Ein kleiner Wert der Entropie bedeutet eine geringe Unreinheit, eine große Entropie eine große Unreinheit der Partition.

Welcher Informationsgewinn durch einen Split erreicht wird, inwieweit also eine Reduktion der Entropie durch die Attributauswahl erzielt wird. Der Informationsgewinn des Attributes A mit der Partition P1,P2...,Pm bezüglich der Startpartition P ist definiert als

$$InfGain(P, A) = Entropie(P) \sum_{i=1}^{m} (\frac{|pi|}{|p|} \cdot Entropie(Pi)) \tag{3.9}$$

Somit wird jeweils das Attribut für den nächsten Split ausgewählt, das den höchsten Informationsgewinn zur Folge hat [19].

Gini-Index

Der Gini-Index ist ein Maß für Ungleichverteilung. Er beschreibt die Abweichung von der vollkommenen Gleichverteilung. Beträgt der Gini-Index 0 ist keine Unreinheit gegeben, nimmt er den Wert 0,5 an, ist die Unreinheit sehr groß. Das Ziel ist somit, einen möglichst kleinen Wert zu erreichen. Die Definition des Gini-Index einer Partition P lautet

$$Gini(P) = 1 - \sum_{i=1}^{k} (pi^2) \tag{3.10}$$

Auch hier ist die Bewertung der Aufteilung des Baumes durch das Split-Attribut relevant.

Der Gini-Index des Split-Attributes in Bezug auf die Start-Partitionierung definiert sich durch den gewichteten Durchschnitt der Gini-Indizes der m Teilmengen.

$$Gini(P, A) = \sum_{i=1}^{m} (\frac{|pi|}{|p|} \cdot Gini(pi)) \tag{3.11}$$

Ein Problem des Entscheidungsbaum-Verfahrens ist die Überanpassung (engl. Overfitting), dass oft sehr große Entscheidungsbäume entstehen können. Dieses Problem folgt aus der Überanpassung (Overfitting) des Entscheidungsmodells. Darum wird oft nach dem Lernen des Entscheidungsbaums ein weiterer Schritt durchgeführt: das sogenannte Vereinfachen (engl. Pruning) des Baums.

3.2.3 Random Forest

Ein Random Forest ist ein Ensemble-Modell, das eine Reihe von Entscheidungsbaum-Klassifikatoren auf verschiedene Teilstichproben (engl. sub-samples) des Datensatzes anpasst und Mittelungen verwendet, um die Vorhersagegenauigkeit und die Kontrolle der Überanpassung zu verbessern. Die Größe der Teilstichprobe ist immer dieselbe wie die ursprüngliche Eingabestichprobengröße, aber die Stichproben werden mit Ersetzung gezogen (bootstrap samples) [26].

In Random Forest werden alle Bäume parallel trainiert (Bagging-Modell/Bootstrap-Aggregation). Darüber hinaus wird jeder Baum im Ensemble aus einem Sample erstellt, das mit Ersetzung (d. h. einem Bootstrap-Sample) aus dem Trainingssatz gezogen wurde. Auch beim Aufteilen eines Knotens während der Konstruktion des Baums ist die gewählte Aufteilung nicht mehr die beste Aufteilung unter allen Features. Stattdessen ist die gewählte Aufteilung die beste Aufteilung unter einer zufälligen Teilmenge der Features. Somit ist die in einem Random Forest eingeführte Zufälligkeit sowohl auf zufällige Samples von Daten als auch auf zufällige Auswahl von Merkmalen beim Aufteilen von Knoten in jedem Baum zurückzuführen. Aufgrund der Mittelwertbildung nimmt die Gesamtvarianz des Modells ab und daher erhält man ein insgesamt besseres Modell.

Beim Erstellen einer Zufallsgesamtstruktur kann man sowohl für die Basisentscheidungsbäume als auch für die Gesamtstruktur spezifische Modellparameter festlegen. Für die Bäume hat man normalerweise die gleichen Parameter wie ein normales Entscheidungsbaummodell wie Baumtiefe, Anzahl der Blätter, Anzahl der Features in jeder Aufteilung, Stichproben pro Blatt, Kriterien für die Knotenaufteilungen, Informationsgewinn und Gini-Verunreinigung. Für die Gesamtstruktur kann man die Gesamtzahl der benötigten Bäume, die Anzahl der pro Baum zu verwendenden Features usw. einstellen [26].

Boosting

Beim Boosting werden iterativ mehrere Modelle desselben Typs erstellt, die aufeinander aufbauen. Jedes neu erstellte Modell ist von der Prognosegüte seines Vorgängers abhängig. Den fehlerhaft klassifizierten Datenobjekten des Vorgängermodells wird ein höheres Gewicht zugewiesen, so dass der Trainingsdatensatz bei jeder Iteration modifiziert wird [9].

3.2.4 Support Vector Machine

Bei der Stützvektormethode (engl. Support Vector Machine (SVM)) werden Daten-objekte wie in vielen anderen Klassifikationsverfahren als Vektoren im d-dimensionalen Datenraum R

repräsentiert. Eine SVM sucht nach einer trennenden Grenze (engl. Margin), die die Punkte mit verschiedener Klassenzugehörigkeit möglichst gut trennt (Siehe Abbildung 3.4).

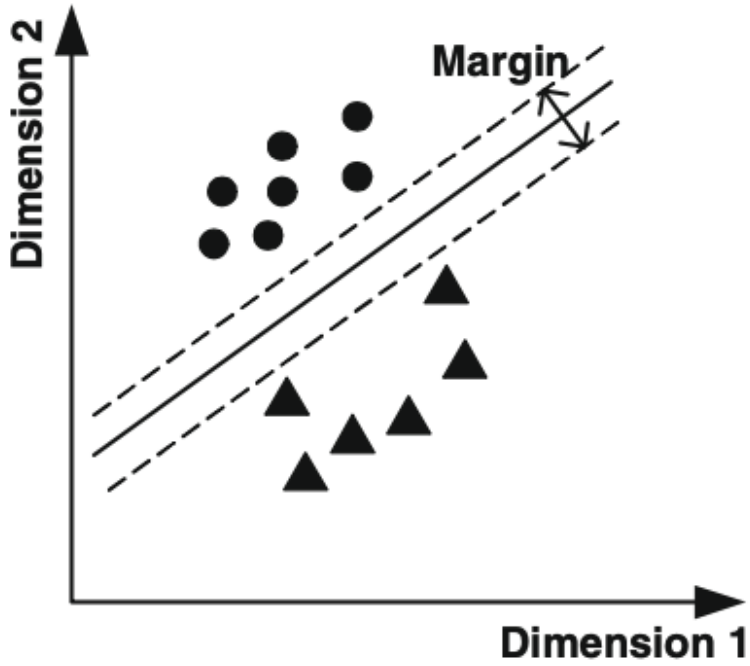


Abbildung 3.4: Support Vector Maschine trennt mithilfe einer trennenden Grenze (Margin) Punkte zweier Klassen (Kreise, Dreiecke)
[19]

Diese Grenze wird durch so genannte Stützvektoren (engl. Support Vectors) repräsentiert. Datenobjekte haben jedoch typischerweise mehr als zwei Attribute (d > 2) und somit ist die Grenze eine d-dimensionale Hyperebene im R. Zusätzlich wird der ursprüngliche Merkmalsraum in einen höherdimensionalen Raum abgebildet, denn in einem höherdimensionalen Datenraum lassen sich möglicherweise eher trennende Grenzen zwischen den Datenobjekten finden. Dieses Verfahren nennt man auch kernel-basiertes Lernen. Ein Kern (engl. Kernel) ist die Transformationsfunktion von Datenobjekten in einen höherdimensionalen Raum. Mathematisch gesehen basiert der Kernel-Trick auf dem Satz von Mercer. Es werden Skalarprodukte im höherdimensionalen Merkmalsraum durch Kernelfunktionen im ursprünglichen Merkmalsraum ersetzt. Häufig verwendet werden beispielsweise die Kern-Funktionen polynomieller Kernel, radialer Basisfunktionskernel und Gauß-Kernel [23].

Die Vorteile der Support Vector Machine liegen in der hohen Klassifikationsgüte bei korrekt spezifiziertem Kernel und der schnellen Klassifikation neuer Datenobjekte. Die Nachteile liegen darin, dass für jeden neuen Datensatz ein erneutes Training erforderlich ist, da die Kern-Funktion spezifiziert wird und deren Parameter geschätzt werden [19].

3.2.5 Künstliche Neuronale Netze

Künstliche Neuronale Netze sind Algorithmen, die dem menschlichen Gehirn nachempfunden sind. Dieses abstrahierte Modell von verbundenen künstlichen Neuronen, ermöglicht es, komplexe Aufgaben aus den Bereichen Statistik, Informatik und Wirtschaft durch Computer zu lösen. Neuronale Netze sind ein sehr aktives Forschungsgebiet und gelten als Grundlage für die künstliche Intelligenz. Durch Neuronale Netze lassen sich verschiedene Datenquellen wie Bilder, Geräusche, Texte, Tabellen oder Zeitreihen interpretieren und Informationen oder Muster extrahieren, um diese auf unbekannte Daten anzuwenden. So lassen sich datengetriebe Vorhersagen für die Zukunft erstellen.

Folgende Definition wird für diese Arbeit übernommen: „Ein (künstliches) neuronales Netz ist ein (gerichteter) Graph G = (U,G), dessen Knoten u $\in$ U Neuronen (engl.neurons, units) und dessen Kanten c $\in$ C Verbindungen (engl.connections) heißen. Die Menge der Knoten ist unterteilt in die Menge Uin der Eingabeneuronen (engl. input neurons), die Menge Uout der Ausgabeneuronen (engl. output neurons) und die Menge U$hidden$ der versteckten Neuronen (engl. hidden neurons) [2].

Für diese Arbeit wird mit mehrschichtigen Perzeptren gearbeitet, die für das Verarbeiten komplexer Informationen geeignet sind (siehe Abb. 3.5).

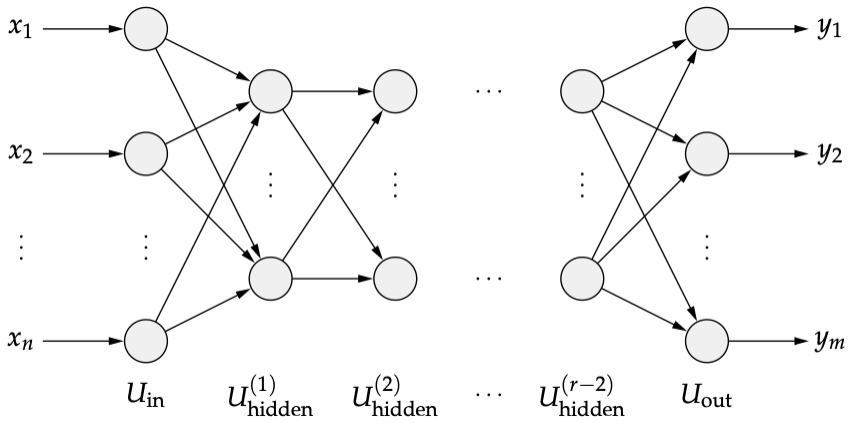


Abbildung 3.5: Allgemeiner Aufbau eines r-schichtigen Perzeptrons
[2]

Anschaulich bedeuten die Einschrankungen des Graphen, dass ein mehrschichtiges Perzeptron aus einer Eingabe- und einer Ausgabeschicht (den Neuronen der Mengen Uin bzw. $Uout$) und keiner, einer oder mehreren versteckten Schichten (den Neuronen in den Mengen U$hidden$) besteht. Die gewichteten Verbindungen innerhalb des neuronalen Netzes bestehen jeweils nur zwischen Neuronen aufeinanderfolgender Schichten. Jedem Neuron sind drei Zustandsfunktionen zugeordnet: Netzeingabefunktion (Propagierungsfunktion), Aktivierungsfunktion und Ausgabefunktion. Die einfachste Variante der Funktionen ist ein zweiwertiger Aktivierungszustand (aktiv oder nicht aktiv).

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{wenn net} \geq \theta, \\ 0, & \text{sonst.} \end{cases}$$

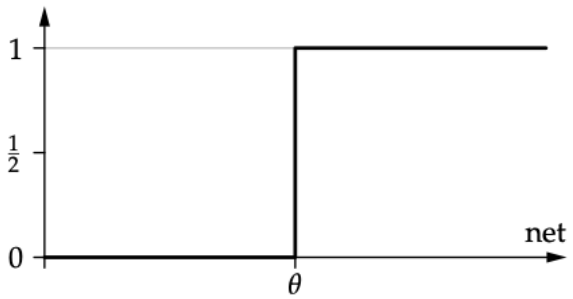


Abbildung 3.6: Sprungfunktion
[2]

Das Neuron wird in diesem Fall nur dann aktiviert, wenn ein zuvor festgelegter Schwellenwert überschritten wird. Als Aktivierungsfunktion wird die Sprungfunktion verwendet (siehe Abb. 3.6).

3.2.6 *k*-Nächster-Nachbar

Ein einfach strukturiertes Klassifikationsverfahren ist der k-Nächster-Nachbar- Klassifikator (engl. k-nearest neighbor). Er erechnet in der Trainingsphase keine explizite Klassifikationsvorschrift, sondern führt lediglich eine Speicherung der Daten durch. Die Auswertung der Daten erfolgt erst, wenn ein neuer Merkmalsvektor klassifiziert werden soll. So ein Verfahren wird auch faules Lernen (engl. Lazy Learning) genannt [23]. Neue Datenpunkte werden dann nach der Entfernung des nächsten Nachbarn klassifiziert. Es speichert alle verfügbaren Fälle in einem Datensatz und klassifiziert neue Datenpunkte weiter mit einem Ähnlichkeitsmaß mit seinen nächsten Nachbarn. Die Ähnlichkeit wird mit einer Entfernungsmessung zwischen dem neuen Datenpunkt und seinen nächsten Nachbarn definiert. Ein beliebtes Distanzmaß ist die Ëuklidische Distanz"[19].

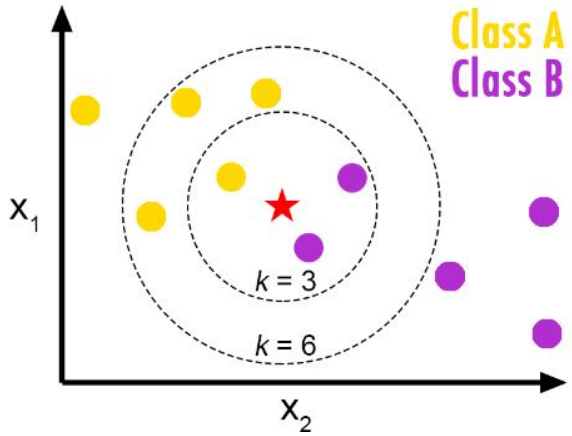


Abbildung 3.7: k-Nächster-Nachbar-Klassifikator
[1]

Zu den größten Vorteilen des Nächster-Nachbar- und Nächste-k-Nachbarn-Klassifikators gehören, dass in der Trainingsphase kein Lernaufwand anfällt, dass der Klassifikator leicht um zusätzliche Trainingsdaten ergänzt werden kann und dass nichtlineare Klassengrenzen gefunden werden können. Zu den größten Nachteilen gehört der hohe Rechenaufwand bei der Klassifikation, da die Unähnlichkeiten mit allen Trainingsvektoren berechnet werden müssen [23].

4 Datensätze

Das folgende Kapitel präsentiert den Datensatz, der in dieser Arbeit verwendet wird. Zuerst werden grundlegende Merkmale des Dateninhaltes beschrieben. Danach wird erklärt, warum gerade dieser Datensatz für diese Arbeit ausgewählt wurde, gefolgt vom tieferen Einblick in qualitative und quantitative Merkmale des Datensatzes.

4.1 Daten Beschreibung

Der ausgewählte Datensatz wurde von [12] veröffentlicht und infolge von der Kaggle-Online-Community extrahiert, die hauptsächlich als eine Wettbewerbsplattform für Data Science und Machine Learning bezeichnet wird. Dieser Satz enthält 17,029 Spam-, und Ham-E-Mails, die aus den Archiven der Linguist-Liste abgerufen wurden. Dabei besitzen die gebundenen E-Mails insgesamt 5 Merkmale (Features).

	Text	Target	Category	Cleaned_Text	Words_Count
0	b'Subject: nesa / hea \' s 24 th annual meetin...	0.0	ham	subject nesa hea th annual meet saddl nesa hea...	206
1	b'Subject: meter 1431 - nov 1999\r\ndaren - \r\...	0.0	ham	subject meter nov daren pleas resolv issu howa...	62
2	b"Subject: investor here .\r\nfrom : mr . rich...	1.0	spam	subject investor mr richard mayer dear friend ...	197
3	b"Subject: hi paliourg all available meds . av...	1.0	spam	subject hi paliourg avail med avail binaur cvo...	176
4	b'Subject: january nominations at shell deer p...	0.0	ham	subject januari nomin shell deer park fyi requ...	121
...	...	...	...	...	...
17024	b'Subject: congratulations ! ! ! ! you have wo...	1.0	spam	subject congratul dayzer intern promot program...	236
17025	b"Subject: new pharm site new great prices hum...	1.0	spam	subject pharm site great price humberto refil ...	101
17026	b"Subject: do you remember us ?\r\n- easily lo...	1.0	spam	subject rememb us easili lose bodi weight incr...	117
17027	b'Subject: pre - approved application sun , 07...	1.0	spam	subject pre approv applic sun nov hello sent e...	115
17028	b"Subject: stop the inexpensive , get med ' ci...	1.0	spam	subject stop inexpens med cine vlcodin inexpns...	20

Abbildung 4.1: Die ersten und letzten vier Zeilen des Datensatzes.

Abbildung 4.1 stellt die Grundstruktur des Datensatzes dar. Die Spalte *Text* zeigt den ersten Teil des Hauptinhalts der E-Mail an. Während die Spalte *Target* bzw. *Category* angibt, ob gerade die E-Mail spam oder ham ist. Dabei steht 0.0 für Ham-E-Mails und 1.0 für Spam-E-Mails. Die *Cleaned-Text* Spalte enthält die bereinigte E-Mails nach Durchführung der Vorverarbeitungs-Schritte. Ein weiteres textbasiertes Merkmal stellt die Spalte *Word-Count* dar, die die Anzahl der Wörter pro E-Mail gibt.

4.2 Datenexploration

Die meisten Klassifikatoren neigen dazu, bei unausgeglichenen Datensätzen schlecht zu funktionieren [4], da Instanzen der unterrepräsentierten Klasse wahrscheinlich ignoriert werden. Der Grund für die Auswahl und Verwendung der Enron-Gruppe ist, dass in diesem Datensatz genügend Spam-Emails vorhanden sind. Die Gesamtzahl der E-Mails in diesem Datensatz beträgt 17029, davon sind 9533 Ham- und 7496 Spam-E-Mails. Abbildung 4.4 zeigt die Verteilung der Spam- und Ham-E-Mails. Es ist ersichtlich, dass der Datensatz zwischen Ham- und Spam-E-Mails ziemlich ausgewogen ist.

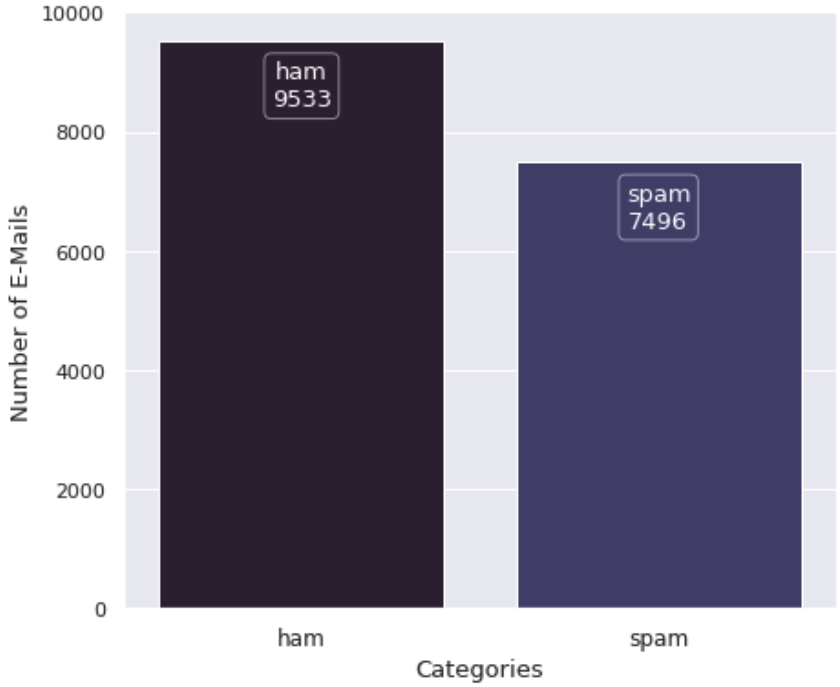


Abbildung 4.2: Verteilung des Datensatztes

Wort Cloud (engl. Word cloud) visualisiert die häufigsten Wörter im gesamten Datensatz, die oft in einem E-Mails-Kontext verwendet werden, wie beispielsweise *subject, messag* und *mail* (Abb 4.3).

Abbildung 4.3: Word Cloud

Wie in Abbildung 4.4 angezeigt, beträgt die durchschnittliche Länge der enthaltenen E-Mails 731 Wörter. Es ist auch erkennbar, dass es eine große Anzahl von E-Mails gibt, die eine Wortzahl von 0 besitzen. Dies weist auf interaktive Merkmale hin, die keine echten “Wörter” im herkömmlichen Sinne haben.

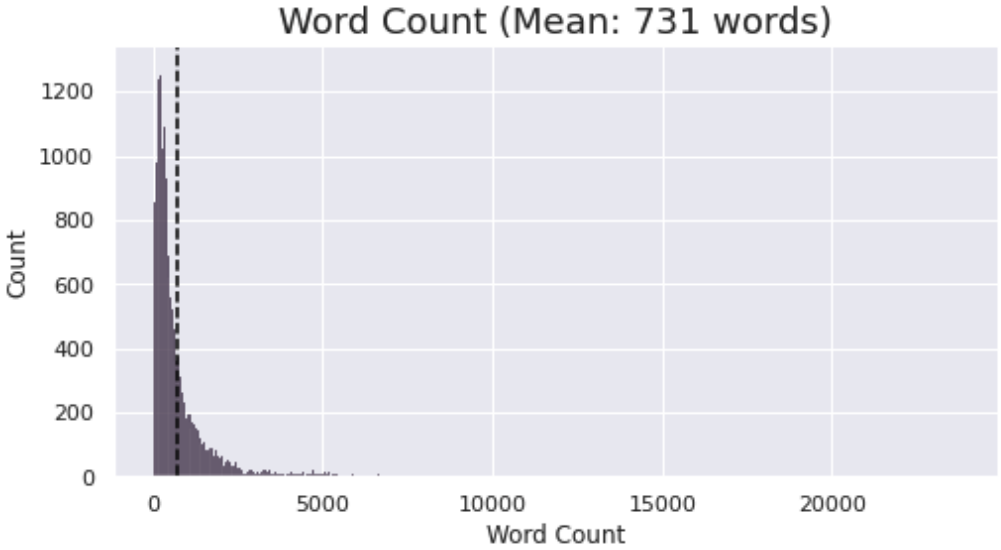


Abbildung 4.4: Wortzahlverteilung

5 Experimente

Basierend auf dem Hauptziel dieser Arbeit, unterschiedliche Ansätze zur Erkennung von Spam-Emails, wird eine Reihe von Experimenten durchgeführt. Insofern gibt dieses Kapitel einen Überblick über das experimentelle Konzept und Design. Darüber hinaus werden die technischen Details zu den verwendeten Python- Bibliotheken, -Tools und -Modellen sowie die konkrete Umsetzung dieser Experimente näher beschrieben.

5.1 Versuchsaufbau

Alle Experimente zielen darauf ab, Kombinationen verschiedener Feature-Extraktionsmethoden und Performance-Metriken zu vergleichen, um zu untersuchen, wie ein Algorithmus eine bessere Leistung im praktischen Sinne als ein anderer erbringen kann. Daher folgt jeder Test demselben systematischen Workflow, der in Abbildung 5.1 dargestellt ist, und in die folgenden Schritte weiter unterteilt wird.

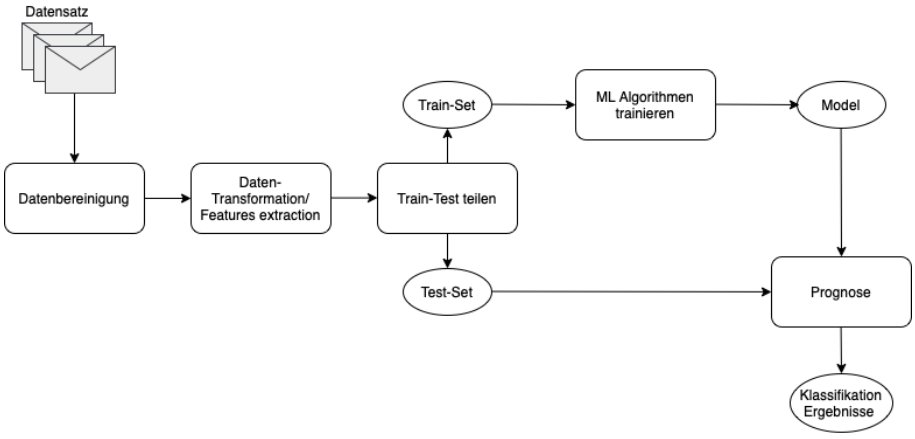


Abbildung 5.1: Experiment-Pipeline.

Datenbereinigung

Bevor Rohdaten durch ein maschinelles Lernmodell gesendet werden können, müssen sie einem Vorverarbeitungschritt unterzogen werden. Das liegt daran, dass Daten in der realen Welt im Allgemeinen unvollständig und inkonsistent sind. Die Qualität der Daten und die daraus ableitbaren nützlichen Informationen haben einen direkten Einfluss auf die Fähigkeit der Modelle zur Verarbeitung dieser Daten. Daher werden im ersten Schritt die rohen Daten der Emails einem Datenbereinigungsmodul übergeben. Dies umfasst Folgendes:

- Umwandlung aller Texte in Kleinbuchstaben
- Entfernen von Stop Wörtern und bestimmten Wörtern wie of, a, cc, ect
- Entfernen von Interpunktion, Sonderzeichen und URLs unter anderen Filtern für reguläre Ausdrücke
- Tokenisierung der Textdaten von Texten zu Wörtern
- Stemmatisieren, um verschiedene Wörter auf dasselbe Wurzelwort zu reduzieren

Datentransformation

Ab diesem Schritt wird ein Vektorraum-Darstellungsmodell für unstrukturierte Daten verwendet, um die Textdaten in numerische Vektoren in einem geeigneten Vektorraum umzuwandeln. Im Rahmen dieser Arbeit wurden zwei verschiedene Prozessen gewählt und ausgeführt: TF-IDF und Word2Vec.

Es wurde eine einfache Implementierung von TF-IDF verwendet, die von Scikit-learn-Bibliothek zur Verfügung gestellt ist. Für die Word2Vec-Einbettung wird ein vorab trainiertes Modell von Google verwendet. Dieses Modell enthält 300-dimensionale Vektoren für 3 Millionen Wörter und Phrasen, die auf 100 Milliarden Wörtern aus dem GoogleNews-Datensatz trainiert wurden.

TF-IDF Prozess wurde ausgewählt, da es eine effektive und weit verbreitete Technik im Information Retrieval und Text Mining ist. Aufgrund der inhärenten Natur des Modells jedoch, das nur eine Tüte unstrukturierter Wörter ist, verliert man zusätzliche Informationen wie Reihenfolge und Kontext um benachbarte Wörter in jedem Textdokument.

Word2Vec hingegen ist eine worteinbettende Darstellung eines Wortes, die mehr Informationen wie Semantik und Struktur über ein Wort als nur eine One-Hot-Darstellung des Wortes erfasst.

Trainings- und Testdaten

Für die Bewertung eines Klassifikators ist es wichtig, dass nicht nur eine hohe Klassifikationsqualität hinsichtlich der in seinem Lernprozess verwendeten Daten (Trainingsdaten), sondern auch bei neuen Datenobjekten gegeben ist. Andernfalls ist der Klassifikator bezüglich der Trainingsdaten überangepasst (Overfitting). Aus diesem Grund ist es wichtig, den vorhandenen Datensatz prozentual aufzuteilen und nur eine Teilmenge der Daten zu lernen. Mit Hilfe der anderen Teilmenge wird die Allgemeingültigkeit des Modells überprüft und es erfolgt eine objektive Bewertung. Die zur Überprüfung des Modells verwendete Teilmenge wird als Testmenge bezeichnet.

Das prozentuale Verhältnis der Trainingsmenge zur Testmenge beträgt 75:25. Damit ein objektiver Vergleich der einzelnen Klassifikatoren möglich ist, wird für alle verwendeten Methoden in dieser Arbeit die gleiche Zerlegung des Datensatzes verwendet.

Klassifikationsalgorithmen

Abschließend werden die im Kapital 3.2 erläuterten Algorithmen (Naive Bayes, Entscheidungsbaum, Random Forest, Support Vector Machine, Künstliche Neuronale Netze und k-Nächster-Nachbar-Classifier) angewendet. Dabei werden alle Algorithmen zuerst auf dem Trainingsdatensatz trainiert und anschließend auf dem Testdatensatz angewendet. Daraufhin folgt die Evaluierung der Modelle anhand der Konfusionsmatrix und der Bewertungsmaße, die im Kapitel 6 aufgeführt und beschrieben werden.

5.2 Technische Implementierung

Python ist eine Open-Source-Sprache, die sowohl in Linux- als auch in Windows-Umgebungen ausgeführt werden kann. Außerdem ist Python eine der beliebtesten und am weitesten verbreiteten Programmiersprachen weltweit. Geschwindigkeit und Komfort sind dabei von zentraler Bedeutung. Die Syntax von Python ist prägnant und dennoch natürlich und verständlich. Zudem enthält sie viele eingebaute Bibliotheken, womit verschiedene NLP-Aufgaben ausgeführt werden können. Darauf aufbauend wurden alle Experimente in Python 3 implementiert, wobei mehrere Bibliotheken verwendet wurden, die allgemein über den Befehl pip auf der Shell installiert werden können.

Python-Bibliotheken

Python ist ein leistungsstarkes Paket, das eine breite Palette von Data Science- und Data Analytics-Anforderungen erfüllt und zahlreiche benutzerfreundlicher Bibliotheken bietet. Im Rahmen dieser Arbeit werden die folgenden Standardbibliotheken von Python verwendet:

Pandas

Pandas ist eine Softwarebibliothek, die bei der Datenanalyse hilft. Es bietet auch eine breite Palette von Funktionen, die sich mit Datenstrukturen und Operationen, wie der Manipulation von numerischen Tabellen und Zeitreihen, befassen.

NumPy

Die NumPy Softwarebibliothek ist ein wesentlicher Bestandteil von Python und wissenschaftlichem Rechnen. Es unterstützt Matrizen mit mathematischen Funktionen auf hoher Ebene. Es unterstützt mehrdimensionale Arrays und kann problemlos in eine Umgebung mit mehreren Datenbanken integriert werden. Es unterstützt auch lineare Algebra, Fourier-Transformationen, Zufallszahlenverarbeitung usw.

NLTK

Das Natural Language Toolkit, oder häufiger NLTK, ist eine Sammlung von Bibliotheken und Programmen für die symbolische und statistische Verarbeitung natürlicher Sprache für Englisch. Es bietet benutzerfreundliche Schnittstellen zu über 50 Sammlungs- und lexikalischen Ressourcen wie WordNet, zusammen mit einer Reihe von Textverarbeitungsbibliotheken für Klassifizierung, Tokenisierung, Wortstammbildung.

Seaborn

Seaborn ist eine Python-Datenvisualisierungsbibliothek, die auf Matplotlib basiert. Es bietet eine High-Level-Schnittstelle zum Zeichnen attraktiver und informativer statistischer Grafiken.

Scikit-learn

Scikit-learn ist wahrscheinlich die nützlichste Bibliothek für maschinelles Lernen in Python. Die Scikit-learn-Bibliothek unterstützt sowohl überwachtes als auch unüberwachtes maschinelles Lernen und enthält viele effiziente Werkzeuge für statistische Modellierung, einschließlich

Klassifizierung, Regression, Clustering und Dimensionsreduktion. Scikit-learn war nötig für das Erstellen von allen Modellen, die in dieser Arbeit verwendet wurden.

Gensim

Gensim ist eine kostenlose Open-Source-Python-Bibliothek für Themenmodellierung, Dokumentenindizierung und Ähnlichkeitsabruf mit großer Sammlung. Ebenfalls ist Gensim ein führendes und hochmodernes Paket zum Verarbeiten von Texten, Arbeiten mit Wortvektormodellen (wie Word2Vec, FastText usw.) und zum Erstellen von Themenmodellen.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse aller Experimente präsentiert. Zunächst werden die als Bewertungsmetriken verwendeten statistischen Maße vorgestellt. Anschließend wird ein Überblick über die Methodenleistungen gegeben.

6.1 Bewertungsmetriken

Das Ziel eines Klassifikationsverfahrens besteht generell darin, unbekannte Datenobjekte möglichst präzise den jeweiligen Klassen zuzuordnen. Zur Modellbewertung existieren verschiedene Gütemaße, die als Vergleich die zu erwartenden Ergebnisse der Datenobjekte benötigen [19].

Klassifikationsgenauigkeit

Die Klassifikationsgenauigkeit (engl. classification accuracy) gibt an, wie viel Prozent der Datenobjekte in der Testmenge korrekt der richtigen Klasse zugeordnet wurden.

$$\text{Klassifikationsgenauigkeit} = \frac{\text{Richtige Klassenzuordnungen}}{\text{Alle Klassenzuordnungen}} \tag{6.1}$$

Klassifikationsfehler

Der tatsächliche Klassifikationsfehler (engl. classification error) gibt analog an, wie viel Prozent der Datenobjekte in der Testmenge einer falschen Klasse zugeordnet wurden:

$$\text{Fehlerrate} = \frac{\text{Falsche Klassenzuordnungen}}{\text{Alle Klassenzuordnungen}} \tag{6.2}$$

Konfusionsmatrix

Eine Konfusionsmatrix (engl. Confusion Matrix) dient zur Beurteilung eines Klassifikators, indem in einer quadratischen Tabelle die Häufigkeiten des Auftretens für alle möglichen Kombinationen von ermittelter Klasse und tatsächlicher Klasse eingetragen werden.

Diese Arbeit befasst sich mit einer binären Klassifikation, weshalb die Gütemaße für binäre Klassifikatoren näher betrachtet werden. Es gibt vier mögliche Kombinationen, die sich aus dem Vergleich der Klassifikation-Ergebnisse mit den erwarteten Werten ergeben. Die Kombinationen werden im Folgenden anhand eines Klassifikators dargestellt, die E-Mails in Spam und Ham einteilt. Die Klassifikation-Qualität bezieht sich auf die Klasse der Spam-E-Mails. Es werden daher vier verschiedene Fälle richtiger und falscher Klassifikationen unterschieden:
TP (richtig positiv): Ein Ham E-Mail wird als Ham klassifiziert.

TN (richtig negativ): Ein Spam E-Mail wird als Spam eingestuft.

FP (falsch positiv): Ein Spam E-Mail wird als Ham eingestuft.

FN (falsch negativ): Ein Ham E-Mail wird als Spam klassifiziert.

Die Ergebnisse werden in einer Konfusionsmatrix dargestellt (Abb 6.1)

		Realität		
		Klasse 1 (positiv)	Nicht Klasse 1 (negativ)	Σ
Klassifikations-Entscheidung	Klasse 1 (positiv)	Richtig Positiv (RP)	Falsch Positiv (FP)	(RP+FP)
	Nicht Klasse 1 (negativ)	Falsch Negativ (FN)	Richtig Negativ (RN)	(FN+RN)
	Σ	(RP + FN)	(FP + RN)	(RP + FN + FP + RN)

Abbildung 6.1: Konfusionsmatrix
[19]

Eine wichtige als Spam gekennzeichnete Email ist eventuell wesentlich unangenehmer als eine nicht als Spam erkannte Spam-Email, in diesem Fall ist eine Gewichtung der beiden Fehler (FP,FN) anhand von (Opportunitäts-) Kosten sinnvoll [19]. **Precision** und **Recall** sind weitere relevante Messgrößen der Klassifikationsgüte, die zwischen beiden Fehlerarten unterscheiden.

Präzision

Präzision (auch Relevanz) (engl. Precision) gibt den Anteil der richtig positiv Klassifi- zierten (RP) an der Gesamtzahl der positiv Klassifizierten (TP + FP) an:

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

Sensitivität

Sensitivität (auch Trefferquote) (engl. Recall) gibt den Anteil der richtig positiv Klassifizierten (TP) an der Gesamtzahl der Positiven (TP + FN) an:

$$Recall = \frac{TP}{TP + FN} \tag{6.4}$$

Präzision und Sensitivität werden insbesondere zur Evaluation von Suchergebnissen beim Information Retrieval herangezogen. Eine Suchmaschine macht im Prinzip für eine Suchanfrage eine Klassifikation von Seiten nach „relevant“ (Treffer) und „nicht relevant“ (kein Treffer). Anhand eines Testdatensatzes kann man analog die Güte einer Suchmaschine messen. Präzision ist dabei der Anteil der relevanten Treffer an allen Treffern. Sensitivität (Recall) ist der Anteil der relevanten Treffer an allen relevanten Seiten [19].

F1-Score

Wenn einer der Werte von **Recall** oder **Precision** gegen Null tendiert, verringert sich auch der Aussagewert des Klassifikators. Daher wird ein f1-Maß für die Mittelung beider Werte definiert und zwar als harmonische Mittelwert der Genauigkeit und der Vollständigkeit:

$$F1 = \frac{2 \times (Precision + Recall)}{Precision \times Recall} \tag{6.5}$$

Die Fläche unter der Kurve

Die Fläche unter der Kurve (eng. Area Under Curve (AUC)) ist eine der am häufigsten verwendeten Metriken zur Bewertung. Es wird für binäre Klassifizierungsprobleme verwendet. Die AUC eines Klassifikators ist gleich der Wahrscheinlichkeit, dass der Klassifikator ein zufällig ausgewähltes positives Datenobjekt höher einstuft als ein zufällig ausgewähltes negatives Datenobjekt.

Falsch-Positiv-Rate und **Richtig-Positiv-Rate** (Achsen des AUC Diagramms) haben beide Werte im Bereich [0, 1]. FPR und TPR werden beide mit unterschiedlichen Schwellenwerten

wie (0,00, 0,02, 0,04, …, 1,00) berechnet und ein Diagramm wird gezeichnet. AUC ist die Fläche unter der Kurve des Diagramms Falsch-Positiv-Rate vs. Richtig-Positiv-Rate an verschiedenen Punkten in [0, 1].

Falsch-Positiv-Rate beschreibt die Wahrscheinlichkeit, dass eine Ham-E-Mail als Spam eingestuft wird und lässt sich wie folgt berechnen:

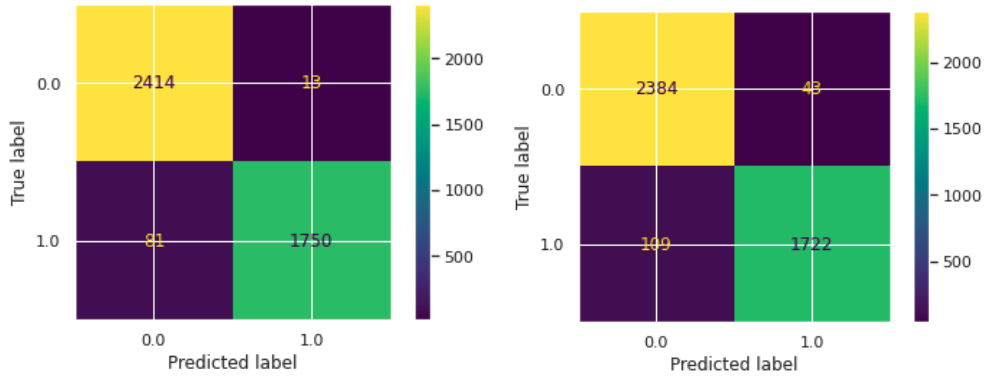
$$Falsch - Positiv - Rate = \frac{FN}{TP + FN} \tag{6.6}$$

6.2 Methodenleistungen

Abbildung 6.2 stellt die Ergebnisse der Konfusionsmatrix für alle Algorithmen dar, die im Rahmen dieser Arbeit mit Tf-idf Feature durchgeführt wurden. Die Algorithmen (Naive Bayes, Random Forest, Support Vector Mashine, Neural network und K-Nearest Neighbours) haben die E-Mails mit Tf-idf Feature sehr effizient bearbeitet und eine Genauigkeit zwischen 0.96 und 0.99 erzielt. Decision Tree 6.2d hingegen hat ca. 1300 Ham-E-Mails als Spam klassifiziert und die Genauigkeit lag bei 0.68.

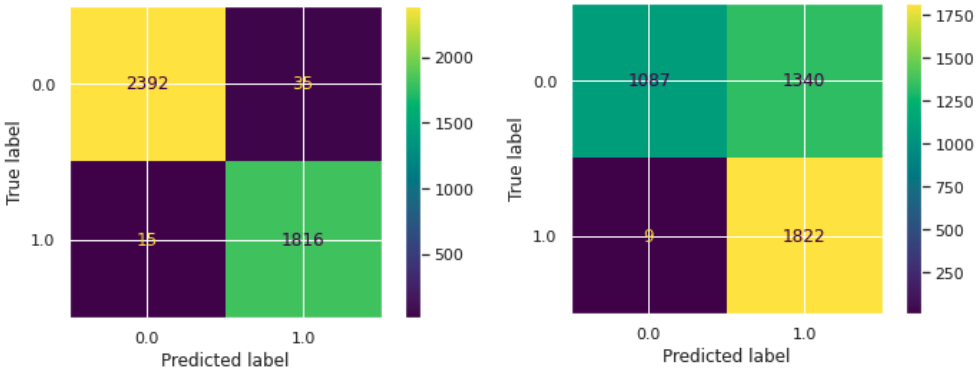
Abbildung 6.3 zeigt ebenfalls die Ergebnisse der Konfusionsmatrix für alle Algorithmen allerdings mit Word2Vec Feature. Alle verwendeten Algorithmen haben die Ham- bzw. Spam-E-Mails richtig eingestuft und eine Genauigkeit zwischen 90 und 99 erzielt, bis auf Naive Bayes Algorithmus, der hingegen ca 1200 Spam E-Mail als Ham klassifiziert hat und die Genauigkeit lag bei 0.72.

Wie bereits erwähnt wurde, je höher die AUC, desto besser ist die Leistung des Modells bei der Unterscheidung der Klassen. Dementsprechend, wenn AUC = 1, dann ist der Klassifikator in der Lage, zwischen allen positiven und negativen Klassenpunkten korrekt zu unterscheiden. Wäre die AUC jedoch 0 gewesen, würde der Klassifikator alle Negative als Positiv und alle Positive als Negativ vorhersagen. Zur Veranschaulichung der Testergebnisse kann in der folgenden Abbildungen 6.4 und 6.5 eine Area Under Curve (AUC) für die Ergebnisse aller Klassifikatoren mit Tf-idf Feature eingesehen werden.



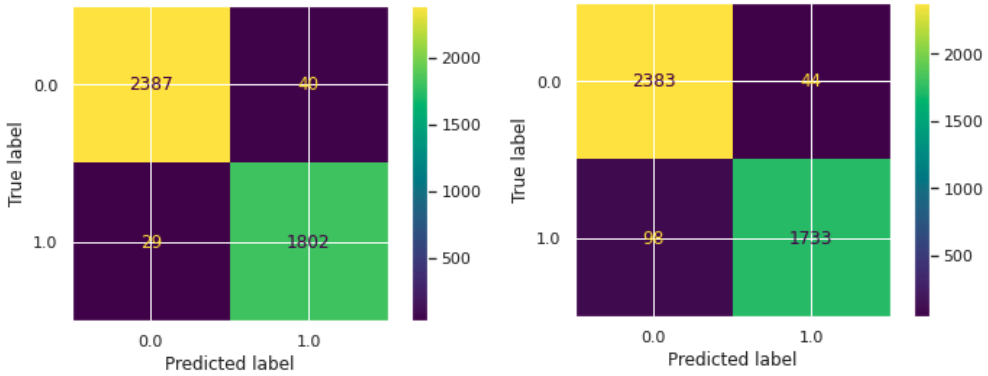
(a) Naive Bayes

(b) Random Forest



(c) Support Vector Mashine

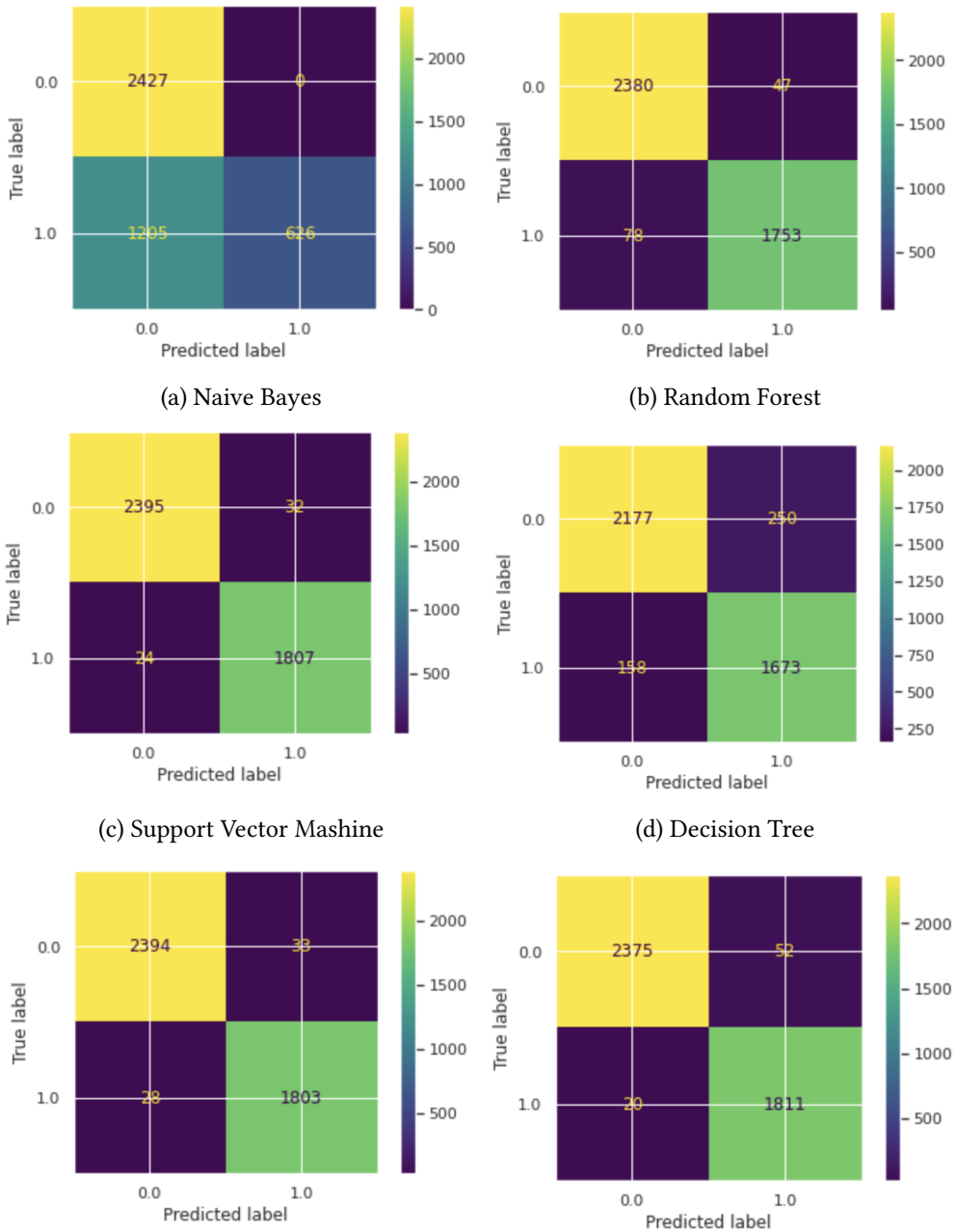
(d) Decision Tree



(e) Neural network

(f) K-Nearest Neighbours

Abbildung 6.2: Konfusionsmatrix aller Algorithmen mit Tf-Idf Feature.



(a) Naive Bayes

(b) Random Forest

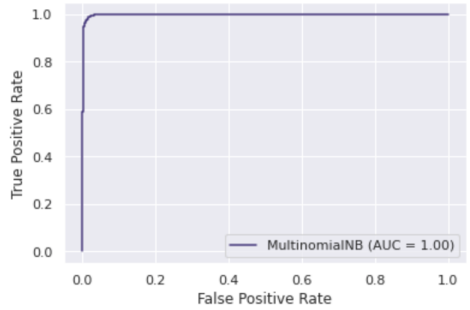
(c) Support Vector Mashine

(d) Decision Tree

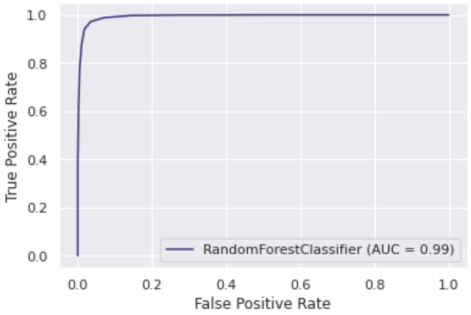
(e) Neural network

(f) K-Nearest Neighbours

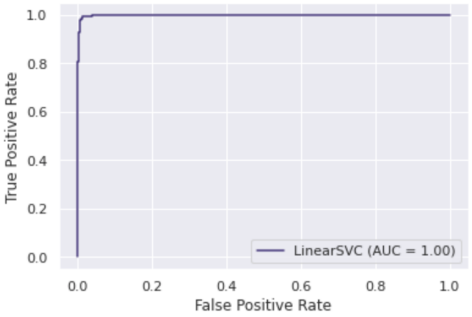
Abbildung 6.3: Konfusionsmatrix aller Algorithmen mit Word2Vec Feature.



(a) Naive Bayes

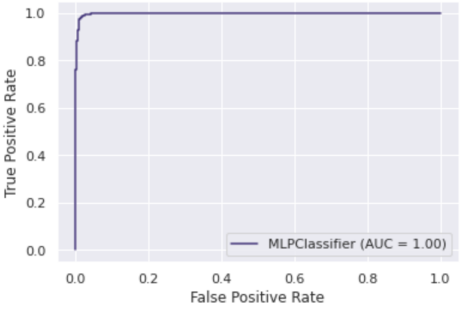


(b) Random Forest

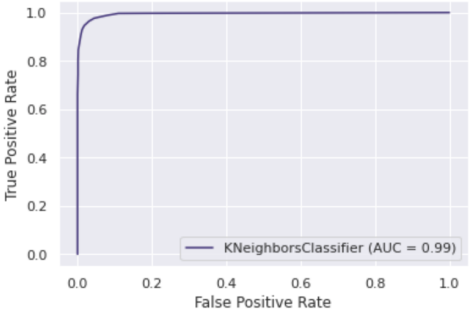


(c) Support Vector Mashine

(d) Decision Tree

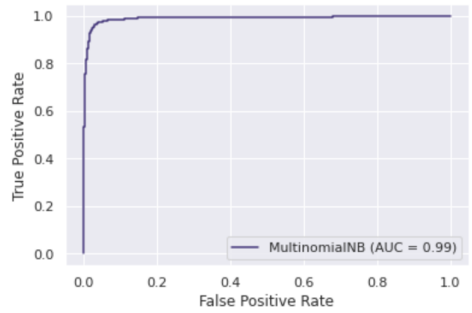


(e) Neural network

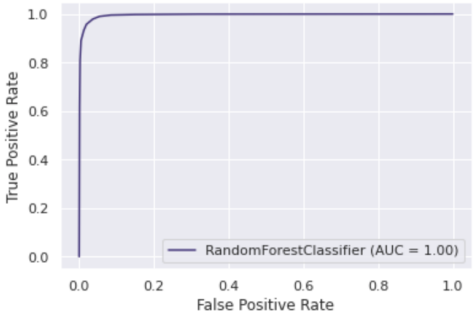


(f) K-Nearest Neighbours

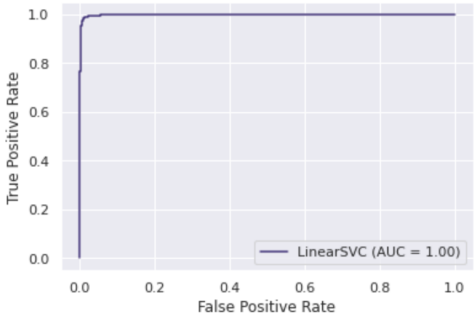
Abbildung 6.4: Area Under Curve (AUC)aller Algorithmen mit Tf-Idf Feature.



(a) Naive Bayes

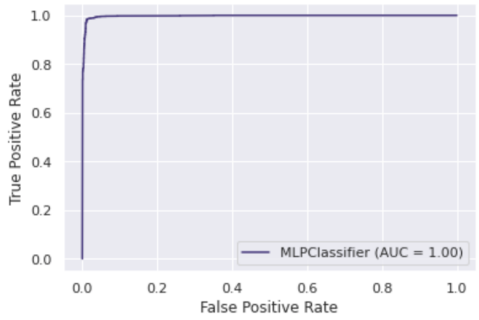


(b) Random Forest

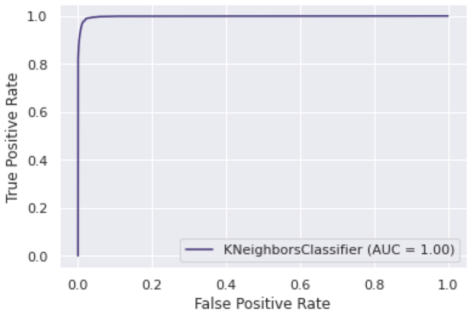


(c) Support Vector Mashine

(d) Decision Tree



(e) Neural network



(f) K-Nearest Neighbours

Abbildung 6.5: Area Under Curve (AUC)aller Algorithmen mit Word2Vec Feature.

Die Tabelle in Abb. 6.6 zeigt die Ergebnisse von sechs überwachten Algorithmen für maschinelles Lernen, zusammen mit zwei verschiedenen Merkmalsextraktionstechniken, Word2Vec und TF-IDF, die auf den Datensatz angewendet werden und anhand von vier Leistungsmetriken (Accurcy, Recall, Precision und F1-Score) gegeneinander bewertet werden.

		Accurcy	Recall	Precision	F1-Score	Time/Sec
TF-IDF	*Naive Bayes*	0.98	0.96	0.99	0.97	9
	Random Forest	0.96	0.94	0.98	0.96	10
	Support Vector Machine	0.99	0.99	0.98	0.99	10
	Decision Tree	0.68	1.00	0.58	0.73	11
	Neural Network	0.98	0.98	0.98	0.98	8
	K-Nearest Neighbours	0.97	0.95	0.98	0.96	3
Word2Vec	*Naive Bayes*	0.72	0.34	1.00	0.51	10
	Random Forest	0.97	0.96	0.97	0.97	20
	Support Vector Machine	0.99	0.99	0.98	0.98	23
	Decision Tree	0.90	0.91	0.87	0.89	24
	Neural Network	0.99	0.98	0.98	0.98	12
	K-Nearest Neighbours	0.98	0.99	0.97	0.98	15

Abbildung 6.6: Leistung von sechs maschinellen Lernalgorithmen.

Wie aus der obigen Tabelle 6.6 ersichtlich ist, haben die meisten Algorithmen eine hohe Genauigkeit unter Verwendung der Tf-idf und Word2Vec Ansätze erzielt. Es ist jedoch nicht sehr hilfreich, nur die Genauigkeit zu betrachten, um die Leistung des Klassifikators zu bewerten. Recall- und Precision-Werte geben einen tieferen Einblick in die Leistung von Klassifikatoren.

Unter Verwendung des Tf-idf-Ansatzes, haben die Klassifikatoren Naive Bayes und Support Vector Machine hohe Präzisions- und Recall-Werte erzielt 0.98 und 0.99. Dies bedeutet, dass sie die meisten Ham- und Spam-E-Mails erkannt und richtig klassifiziert haben. Das Decision-Tree-Algorithmus hingegen hat ebenfalls einen höheren Recall-Wert von 1. Was besagt, dass der Klassifikator die meisten Spam-E-Mails korrekt vorhersagen konnte, jedoch viele der Ham-E-Mails nicht als Ham vorhersagen konnte.

Andererseits erzielten sowohl SVM- als auch neuronale Netzwerkalgorithmen mit dem Word2Vec-Ansatz ebenfalls eine hohe Genauigkeit, benötigten jedoch doppelt so viel Zeit für die Vorhersage im Vergleich zum Tf-idf-Ansatz.

Zudem hat das Naive-Bayes-Algorithmus einen sehr hohen Präzisions-Wert und konnte alle Ham-E-Mails korrekt vorhersagen, was gut aber gleichzeitig gefährlich ist, da er ebenfalls die meisten Spam-E-Malis als Ham eingestuft hat.

7 Abschlussbetrachtung

Das Kapitel fasst die Arbeit zusammen, setzt sich kritisch mit dem Ergebnis auseinander und gibt einen Ausblick auf weitere Verbesserungsmöglichkeiten.

7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde die automatisierte Erkennung von Spam-E-Mails durch maschinelles Lernen untersucht. Dazu wurden zwei Merkmalsextraktionsmethoden Tf-idf und Word2Vec zusammen mit sechs verschiedenen Klassifikatoren (Naive Bayes, Entscheidungsbaum, Random Forest, Support Vektor Machine, Künstliche Neuronale Netze und K-Nächster-Nachbar) kombiniert und an einem öffentlich verfügbaren Filter-Spam-Datensatz getestet. Dieser Datensatz wurde entsprechend der beschriebenen Vorverarbeitungsschritten im Abschnitt 2.4 bereinigt und für die weitere Untersung vorbereitet. Für jede Kombination wurde eine Testpipeline verwendet, um Klassifizierungsergebnisse zu vergleichen und herauszufinden, welcher Algorithmus mit welchem Ansatz bessere Leistung bei der Erkennung von Spam-E-Mails erzielt.

Für die Untersuchung wurden 17029 E-Mails (9533 Ham-E-Mails- und 7496 Spam-E-Mails) im Hinblick auf ihren Inhalt analysiert, davon beträgt die Trainingsmenge 12771 und die Testmenge 4258, die erforderlich ist, für die Allgemeingültigkeit des Modells zu überprüfen. Die Analyse ergab, dass sich zwei Algorithmen (SVM, NN) am besten sowohl mit dem Tf-idf- als auch Word2Vec-Ansatz zur Klassifizierung von Spam-E-Mails eignen. Dabei konnten beide Algorithmen fast alle Ham- und Spam-E-Mails korrekt vorhersagen, mit einer Genauigkeit zwischen 0,98 und 0,99. Allerdings muss berücksichtigt werden, dass die Verarbeitungszeit der Klassifikation mit dem Tf-idf-Ansatz viel kürzer war als mit dem Word2Vec-Ansatz.

7.2 Ausblick

Dieser Arbeit dient als erster Schritt in Richtung des Testens der Leistung Maschinen Learning Algorithmen bei der E-Mail-Klassifizierung, mit dem Ziel der Spam-E-Mail-Filterung.

Spam-E-Mails sind seit langen eine der größten Herausforderungen, denen die Welt sich im Internet gegenüberstehen, und werden es zumindest in naher Zukunft bleiben. Da die Spammer immer ausgeklügelter werden, ist es für die Industrie und die Wissenschaft unerlässlich, mit den neuen Entwicklungen im Spam-E-Mail-Versand Schritt zu halten.

Die weitere Arbeit in dieser Richtung kann jedoch andere Faktoren berücksichtigen, die neben dem Nachrichteninhalt ein starkes Indiz für Spam sind, wie z. B. E-Mail-Header, URLs, bildbasierte Filterung, verhaltensbasierte Filterung, und Analysieren des sozialen Netzwerks der Benutzer durch Analysieren der Felder „From“, „To“, „CC“ und „BCC“ der E-Mail-Nachrichten.

Literaturverzeichnis

[1] Bde: k-Nächster-Nachbar-Klassifikator. (2016). – URL https://www.pm365.tk/ProductDetail.aspx?iid=155049759&pr=39.88

[2] Borgelt, Christian ; Braune, Christian ; Klawonn, Frank ; Moewes, Christian ; Steinbrecher, Matthias: *Computational Intelligence*. Springer, 2015

[3] Carstensen: *Computerlinguistik und Sprachtechnologie*. Springer, 2010

[4] Chowdhury, Gobinda G.: *Natural language processing*. Annual review of informa- tion science and technology, 2003

[5] Fayyad, U. M. ; Piatetsky-Shapiro, G. ; Smyth, P. ; Uthurusamy, R.: *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996

[6] Feldman, Ronen ; Dagan, Ido: Knowledge discovery in textual databases. (1995)

[7] Ganesan, Kavita: Text Preprocessing for Machine Learning NLP. (2021). – URL https://kavita-ganesan.com/text-preprocessing-tutorial/#.YXlIcS221QI

[8] Graham, P.: A Plan for Spam. (2002)

[9] HastieRobert, Trevor ; TibshiraniJerome ; Friedman: *The Elements of Statistical Learning*. Springer, 2009

[10] Hearst, Marti: Text Data Mining. (1997)

[11] Hearst, Marti: What Is Text Mining? (2003)

[12] Junior, Wander F.: Enron-Spam dataset. (2020). – URL https://www.kaggle.com/wanderfj/enron-spam

[13] KRISHNA, Rajan: *Informatics for Materials Science and Engineering*. Elsevier Science, 2013

[14] Luber, Stefan ; Litzel, Nico: Was ist Natural Language Processing? (2016). – URL https://www.bigdata-insider.de/was-ist-natural-language-processing-a-590102/

[15] Lämmel, Uwe ; Cleve, Juergen: *Data Mining*. De Gruyter Oldenbougr, 2016

[16] Maria Giatsoglou, Manolis G. V.: entiment Analysis Leveraging Emotions and Word Embeddings. (2016)

[17] Mikolov, Tomas ; Corrado, Greg ; Chen, Kai ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. (2013)

[18] Mikolov, Tomas ; Sutskever, Ilya ; Chen, Kai ; Dean, Jeffrey ; Corrado, Greg: Distributed Representations of Words and Phrases and their Compositionality. (2013)

[19] Müller ; Lenz, Roland M. ; Hans-Joachim: *Business Intelligence*. eXamen.press, 2013

[20] Pan, S. J. ; Yang, Q.: A Survey on Transfer Learning. In: *in IEEE Transactions on Knowledge and Data Engineering* (2010)

[21] Qaiser, Shahzad ; Ali, Ramsha: Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. In: *International Journal of Computer Applications* (2018)

[22] Rabe, L.: Prognose zur Anzahl der täglich versendeten und empfangenen E-Mails weltweit. (2021). – URL https://de.statista.com/statistik/daten/studie/252278/umfrage/prognose-zur-zahl-der-taeglich-versendeter-e-mails-weltweit/

[23] Runkler, Thomas A.: *Data Mining*. Springer, 2010

[24] S., Guzella T. ; M., Caminhas W.: A review of machine learning approaches to Spam filtering. (2009)

[25] Sahami, M. ; Dumais, S. T. ; Heckerman, D. ; Horvitz, E.: A Bayesian Approach to Filtering Junk E-Mail. (1998)

[26] Sarkar, Dipanjan: *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*. Springer, 2019

[27] Satapathy, Ranjan ; Guerreiro, Claudia ; Chaturvedi, Iti ; Cambria, Erik: Phonetic-Based Microtext Normalization for Twitter Sentiment Analysis. (2017)

[28] Segall, Richard: Text Miner and Megaputer Polyanalyst. (2016)

[29] Simoudis: *Reality check for data mining*. IEEE Expert, 1996

[30] Thatha, Venkata N. ; Babu, A. S. ; Haritha, D.: *Smart Intelligent Computing and Applications- An Enhanced Feature Selection for Text Documents*. Springer, 2020

[31] Wennker, Phil: *Informatics for Materials Science and Engineering*. Springer, 2020

[32] Wiesalla, Luise: KONZEPTE IM MACHINE LEARNING. (2021). – URL https://www.nextlytics.com/de/blog/konzepte-machine-learning

[33] Zhang, L. ; Zhu, J. ; Yao, T.: An evaluation of statistical spam filtering techniques. (2004)

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Erkennung von Spam-Emails mit Ansätzen des Maschinellen Lernens

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original