

Duale Hochschule Baden-Württemberg Mannheim

Seminararbeit

Autorenklassifikation

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser(in):	Anabel Lilja (4279481) Bjarne Gerdes (8608827) Johannes Deufel (5610649) Simone Marx (6147264)
Kurs:	WWI18DSB
Studiengangsleiter:	Prof. Dr. Bernhard Drabant
Wissenschaftlicher Betreuer:	Dr. Manfred Preißendörfer
Bearbeitungszeitraum:	18.11.2020 - 26.01.2020

Ehrenwörtliche Erklärung

Wir versichern hiermit, dass wir die vorliegende Arbeit mit dem Thema: *<Titel Ihrer Arbeit>* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel genutzt haben. Wir versichern zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Anabel Lilja

Ort, Datum

Bjarne Gerdes

Ort, Datum

Johannes Deufel

Ort, Datum

Simone Marx

Inhaltsverzeichnis

Abbildungsverzeichnis	iii
Tabellenverzeichnis	iv
Abkürzungsverzeichnis	v
1 Einleitung	1
1.1 Relevanz von Autorenklassifikation	1
1.2 Zielsetzung und Aufbau der Ausarbeitung	2
2 Theoretische Grundlagen	4
2.1 Ausgewählte Konzepte des Machine Learning	4
2.2 Related Work	6
3 Praktische Umsetzung	9
3.1 Vorverarbeitung	11
3.2 Linear Support Vector Machine	14
3.3 Logistische Regression	16
3.4 Extreme Gradient Boosting	17
3.5 Clustering mittels K-means	19
4 Diskussion der Ergebnisse	20
4.1 Bewertung der Modelle für die Zielvariable Alter	21
4.2 Bewertung der Modelle für die Zielvariable Geschlecht	23
4.3 Bewertung der Modelle für die Zielvariable Sternzeichen	24
4.4 Bewertung der Modelle für die Zielvariable Thema	26
4.5 Bewertung des Clusterings	27
5 Diskussion der Ergebnisse	29
5.1 Fazit	29
5.2 Ausblick	30
A Tabellen	31
A.1 Feature Engineering	31
A.2 Clustering-Features	32
Literaturverzeichnis	33

Abbildungsverzeichnis

Abbildung 3.1 Visualisierung der Vorgehensweise	9
Abbildung 3.2 One Hot Encoder	13
Abbildung 3.3 Standardscaler	14
Abbildung 4.1 Bewertung der Modelle für die Zielvariable Alter	21
Abbildung 4.2 Bewertung der Modelle für die Zielvariable Geschlecht	23
Abbildung 4.3 Bewertung der Modelle für die Zielvariable Sternzeichen	25
Abbildung 4.4 Bewertung der Modelle für die Zielvariable Thema	27
Abbildung 4.5 Validierungsergebnisse des Clusterings	28

Tabellenverzeichnis

Tabelle 2.1	Ergebnisse der Related Work	8
Tabelle 3.1	Zuordnung der Transformatoren zu den verarbeiteten Variablen	14
Tabelle 3.2	Optimale Parameter für die Linear Support Vector Machine (LSVM)	16
Tabelle 3.3	Optimale Parameter für die logistische Regression	17
Tabelle 3.4	Optimale Parameter für den XGBoost	18
Tabelle A.1	Durch das Feature Engineering erarbeitete Features	31
Tabelle A.2	Interpretationen der textuellen und numerisch basierten Cluster	32

Abkürzungsverzeichnis

BoW	Bag of Words
KNN	K-Nearest Neighbors
MSE	Mean Squared Error
NLP	Natural Language Processing
SVM	Support Vector Machine
LSVM	Linear Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
SGD	Stochastic Gradient Descent

1 Einleitung

Die vorliegende Ausarbeitung beschäftigt sich mit der Thematik der Autorenklassifikation im Forschungsfeld des maschinellen Lernens. Im Rahmen dieser Ausarbeitung werden Machine Learning Modelle erstellt, welche anhand eines Input-Textes das Alter, Geschlecht und Sternzeichen eines Autors, sowie das Genre seines Textes voraussagen. Die Vorhersage der Variablen Geschlecht und Alter konnte hier am besten getroffen werden, während die Vorhersage der Variable Sternzeichen die schlechteste Performance erreichte. Hierfür wurden verschiedene Klassifikatoren verwendet und die Vorgehensweisen, die Ergebnisse und die Erkenntnisse dieser miteinander verglichen. Ebenfalls wird die bestehende Literatur dieses Forschungsgebiets untersucht, wodurch die entwickelten Ansätze dieser Ausarbeitung in den aktuellen Forschungsstand eingeordnet werden können.

1.1 Relevanz von Autorenklassifikation

Die Basis dieser Ausarbeitung findet sich in der Wissenschaft der Stilometrie. Die Stilometrie ist eine Disziplin, welche Untersuchungen von Sprachstilen mithilfe statistischer Mittel durchführt. Sie fasst dabei die Annahme auf, dass jeder Autor einen individuellen Schreibstil besitzt, welcher durch die unbewusste aber konsistente Verwendung bestimmter Wortmuster entsteht. Die Verwendung solcher Muster erstreckt sich dabei über alle verfassten Dokumente eines Autors über die Zeit hinweg [11]. Dies führt dazu, dass sich verschiedene Schreibstile anhand statistischer Methoden numerisch abgrenzen lassen und ermöglicht die Zuordnung von Autoren und Texten mithilfe gängiger Indikatoren wie die Wort- und Satzlänge, der Wortschatz, die Häufigkeit von Worten und der Zusammenhang zwischen den verwendeten Wörtern. Es ist ebenfalls möglich, gewisse Eigenschaften von Autoren und Texten zu bestimmen [5, 2].

Die Kenntnis des Profils eines Autors und die Zuordnung zu Texten kann in den unterschiedlichsten Bereichen von hoher Bedeutung sein. So bedient sich die forensische Linguistik den Mitteln und Erkenntnissen der Stilometrie, um beispielsweise das sprachliche Muster einer verdächtigen Textnachricht zu analysieren und dieses mit verdächtigen Personen oder einer Datenbank abzugleichen [15]. Hierbei lassen sich außerdem Rückschlüsse auf bestimmte Charakteristiken wie das Alter, Geschlecht, familiärer und gesellschaftlicher Hintergrund, Umgangs- und Muttersprache oder Interessen ziehen, welche bei polizeilichen Ermittlungen von erheblichem Vorteil sein können [14].

Charakteristiken dieser Art von Einzelpersonen oder Personengruppen sind auch aus Marketingperspektive interessant. Unternehmen benötigen solche Informationen über ihre Kunden sowohl für die Kundenbindung als auch für die Kundengewinnung. Sie ermöglichen unter

anderem die Bereitstellung personalisierter Produkte und Dienstleistungen und das damit einhergehende personalisierte Marketing [9, 16].

Ebenfalls hilfreich und notwendig ist die Autorenklassifikation in der Bibliothekswissenschaft und der Geschichtswissenschaft. Die Zuordnung von Autoren zu Werken, sowie die Klassifikation dieser in bestimmte Genres oder Jahrgänge kann eine große Hilfe bei der Verwaltung von Büchereien und Bibliotheken darstellen [17, 13]. Des Weiteren kann diese Klassifikation Historikern einen hohen Mehrwert bieten, da zum Beispiel unbekannte Werke einer bestimmte Epoche oder einer bestimmten Region zugeordnet werden können, hierdurch können historische Rückschlüsse über diese Zeit oder Gesellschaft gezogen werden. Dies kann auch in der heutigen Zeit einen hohen Wert haben, da Sentimentanalysen und Klassifikationen ebenfalls Rückschlüsse über soziales und psychologisches Verhalten in der Gegenwart bringen können [1].

Aus akademischer Sicht kann die Autorenklassifikation vor allem bei der Erkennung von Plagiaten behilflich sein [18]. Die Übernahme fremder geistiger Leistung verstößt in der Wissenschaft gegen Prüfungsordnungen, Arbeitsverträge oder Universitätsrechte und kann durch Machine Learning Klassifikatoren zur Autorenuordnung leichter erkannt werden.

1.2 Zielsetzung und Aufbau der Ausarbeitung

Das Ziel dieser Ausarbeitung ist es, das Alter, Geschlecht und Sternzeichen eines Autors, sowie das Jahr der Veröffentlichung und das Genre seines Textes anhand eines Textinputs mithilfe unterschiedlicher Machine Learning Modelle zu bestimmen.

Kapitel 2 dieser Ausarbeitung dient der Darstellung der theoretischen Grundlagen, welche im folgenden Kapitel 3 Anwendung finden. Im Zuge dessen wird außerdem die bestehende Literatur zu dieser Thematik analysiert und verschiedene Ansätze beschrieben, sodass aktuelle Vorgehensweisen, Ergebnisse und Erkenntnisse mit unseren verglichen werden können.

Die Modellerstellung erfolgt in Kapitel 3 dieser Ausarbeitung. Zur Erstellung wird der *Blog Authorship Corpus* verwendet, welcher im Jahr 2017 auf der Website der Online-Community Kaggle veröffentlicht wurde. Der Datensatz umfasst insgesamt 681.288 Blogeinträge von 19.320 Verfassern aus dem August 2004.

Die tatsächliche Umsetzung kann in sieben Schritte unterteilt werden. Als erster Schritt wird Feature Engineering betrieben, um Merkmale aus den Rohdaten zu extrahieren. Im Rahmen dieser Ausarbeitung werden 16 Merkmale untersucht (siehe Anhang A1), darunter Aspekte wie die Textlänge, die Anzahl der Emails im Text und der Anteil der Großbuchstaben an allen Zeichen. Diese Merkmale können anschließend verwendet werden, um die Leistung der zu erstellenden Machine Learning Algorithmen zu verbessern.

Der zweite Schritt des Cleaning beinhaltet die Entfernung jeglicher Unreinheiten in den Daten, sodass im weiteren Verlauf mit einem bereinigten Datensatz gearbeitet werden kann. Unreinheiten sind beispielsweise diakritische Zeichen wie Häkchen, Striche und Kringel.

Anschließend werden die Daten im dritten Schritt transformiert, um sie in ein für die Modelle lesbares Format zu bringen. Im Rahmen dieser Ausarbeitung wird dabei zwischen Textdaten und anderen Daten unterschieden und je nach vorhandenem Datentyp eine geeignete Form der Datentransformation vorgenommen.

Anschließend an die Datentransformation werden sowohl supervised als auch unsupervised Machine Learning Verfahren verwendet, um unterschiedliche Modelle zu trainieren und diese mit weiteren Features anzureichern.

Im sechsten Schritt wird Parametertuning angewendet, um die optimalen Parameter für die erstellten Modelle zu finden. Anschließend werden die Modelle im siebten Schritt anhand unterschiedlicher Evaluationsmetriken bewertet und ,gewichtet anhand ihrer Verluste, kombiniert.

Die finalen Modelle werden dann mittels einer flask-API an ein im Rahmen dieser Ausarbeitung entwickeltes Frontend angebunden. Über ein Eingabefeld auf der Website kann ein Text-Snippet bereitgestellt werden, anhand dessen die angebundenen Modelle Vorhersagen über das Alter, Geschlecht und Sternzeichen eines Autors, sowie das Jahr der Veröffentlichung und das Genre des Textes treffen.

Anschließend an die Entwicklung der Modelle beinhaltet Kapitel 4 die Diskussion der Entwicklungsergebnisse. Zum einen wird hier auf die Performance der Modelle eingegangen, welche anhand verschiedener Evaluationsmetriken, wie die Accuracy oder den F1-Score, bewertet werden kann. Die erstellten Modelle werden dabei miteinander, aber auch mit anderen Modellen aus der Literatur verglichen und in den aktuellen Forschungsstand eingeordnet. Zum anderen wird ein besonderer Fokus auf die Analyse der Vorgehensweise und die daraus gewonnenen Erkenntnisse gelegt, da diese einen hohen Mehrwert für zukünftige Arbeiten dieser Art haben können.

Abschließend bietet die Schlussbetrachtung eine Zusammenfassung der Ergebnisse und Erkenntnisse, sowie einen zukunftsorientierten Ausblick über die Entwicklung des Forschungsobjektes.

2 Theoretische Grundlagen

Als Basis für die in Kapitel 3 folgende Erstellung von Machine Learning Modellen zur Klassifikation von Autoren werden in diesem Kapitel ausgewählte theoretische Konzepte des Machine Learnings und der Autorenklassifikation erläutert. Außerdem werden Vorgehensweisen und Forschungsergebnisse aus der bestehenden Literatur untersucht. Hierbei werden sowohl ähnliche, als auch von unserer Vorgehensweise abweichende Methoden beleuchtet.

2.1 Ausgewählte Konzepte des Machine Learning

Test und Train Split:

Ein Datensatz wird üblicherweise in Test- und Trainingsdaten unterteilt. Mit den Trainingsdaten wird ein Modell erstellt, welches bestimmte Parameter erlernt. Die Testdaten werden verwendet, um die Qualität des Modells zu bewerten, indem versucht wird, die Zielvariable der Testdaten mit dem Modell vorherzusagen. Es zeigt also, wie gut das erstellte Modell Vorhersagen auf Daten treffen kann, die nicht im Training verwendet wurden.

Supervised und unsupervised learning:

Supervised und unsupervised learning sind zwei unterschiedliche Lernansätze des Machine Learning. Beim supervised learning handelt es sich um das sogenannte überwachte Lernen. Hier sind die Trainingsdaten mit einem Label versehen, was bedeutet, dass Funktionen auf Basis von bereits bekanntem Output ermittelt werden. Anwendungsfälle für diese Lernart sind Neuronale Netze, Decision Trees und Lineare Regression. Im Gegensatz dazu ist es das Ziel des unsupervised learnings, also des unüberwachten Lernens, aus Daten unbekannte Muster zu erkennen und Regeln aus diesen abzuleiten. Die vorhandenen Daten sind hierbei also nicht mit einem Label versehen. Anwendungsfälle für diese Lernart sind das Clustering und die Density Estimation.

Regression:

Die Regression beschreibt im Bereich des Machine Learning den Versuch, Beziehungen zwischen einer abhängigen und einer oder mehreren unabhängigen Variablen zu modellieren. Ziel ist die quantitative Beschreibung von Zusammenhängen, sowie die Prognose von Werten der abhängigen Variable. Bei den Zielvariablen handelt es sich um kontinuierliche oder numerische Variablen. Bekannte Regression-Methoden sind die Lineare Regression, Lineare Support Vector Machine (SVM) und Regression Trees.

Classification:

Im Bereich des Machine Learning versucht die Classification anhand einer Funktion von Zielvariablenwerten auf diskrete oder kategoriale Zielvariablen zu schätzen. Als Funktion wird in der Mathematik eine Zuordnung definiert, welche jedem Element x aus einer Menge X eindeutig ein Element y aus einer Menge Y zuordnet. Man unterscheidet je nach Anzahl an Zielvariablenwerten in binäre und multivariate Classification. Bekannte Classification-Methoden sind die logistische Regression, Naive Bayes, Decision Trees und K-Nearest Neighbors (KNN).

Clustering:

Das Clustering ist wie bereits beschrieben ein prominentes Beispiel für die Methode des unsupervised learnings. Da hier die Datenpunkte nicht mit einem Label gekennzeichnet sind, müssen die Daten als Ziel des Clusterings strukturiert werden. Hierfür werden im machine learning mithilfe eines Clusteringalgorithmus wie beispielsweise K-means automatisiert bestimmte Datenpunkte aufgrund ihrer Ähnlichkeit gruppiert und einem bestimmten Cluster zugeordnet. Die Anzahl der Cluster gilt es dabei vorab festzulegen. Die verschiedenen Cluster stellen anfangs eine verborgene Variable dar. Erreicht das Clustering sein Ziel, die Variable sichtbar zu machen, kann dem Datensatz diese beispielsweise als neues Feature hinzugefügt und in einem supervised Learning verwendet werden. [12]

Hyperparameter Optimization:

Hyperparameter sind alle Parameter, die vom Benutzer vor Beginn des Trainings willkürlich eingestellt werden können und bestimmen die grundlegende Struktur des Modells. Ziel ist es, die optimale Kombination der Parameterwerte zu finden, sodass entweder der Verlust der Funktion minimal oder die Accuracy der Funktion maximal wird. Dies ist vor allem für den Vergleich der Güte von verschiedenen Modellen auf einen Datensatz hilfreich. Es gibt verschiedene Verfahren zur Optimierung der Hyperparameter, wie beispielsweise Manual Search, Random Search oder Grid Search.

Cross Validation

Die Cross Validation ist eine statistische Methode, die zur Einschätzung der Güte von Modellen des maschinellen Lernens verwendet wird. Dabei wird der Datensatz in k gleichgroße Partitionen unterteilt. Für jede Partition wird die Partition als Testdatensatz und der Rest der Partitionen als Trainingsdatensatz verwendet. Es wird ein Modell an den Trainingsdatensatz angepasst und anhand des Testdatensatzes evaluiert. Die Evaluationsergebnisse der einzelnen Modelle fassen die Güte des Modells zusammen, indem die Gesamtfehlerquote als Durchschnitt aus den einzelnen Fehlerquoten der k Partitionen berechnet wird. Diese Art der Validation ist vor allem bei kleinen Datensätzen von Vorteil, da sie diese optimal nutzt.

Grid Search:

Grid Search ist ein Verfahren zur Optimierung der Hyperparameter eines Machine Learning Modells. Hierbei wird ein Raster an Hyperparameter gebildet. Das Modell wird dann

auf jede mögliche Kombination von Hyperparametern in diesem Raster trainiert und getestet. Die wichtige Komponente ist dabei die Auswahl der Hyperparameter. Hier können beispielsweise die Learning Rate oder der Hinge Loss verwendet werden.

Stochastic Gradient Descent (SGD)

Der SGD minimiert eine Funktion. Über verschiedene Iterationen wird ein neuer Funktionswert, der näher am Minimum liegt berechnet. Dies geschieht, indem die Ableitung der zu minimierenden Funktion an einem gegebenen Punkt multipliziert mit der definierten Learningrate α , die so als „Schrittweite“ von dem vorherigen Punkt abgezogen wird. Die letzte Iteration ist dadurch definiert, dass das Minimum erreicht ist oder nicht besser erreicht werden kann. Dabei kann die Funktion nahezu beliebig viele Feature haben, die dann als Dimensionen bzw. Funktionswerte verstanden werden. Der SGD setzt dabei die Besonderheit um, dass nur ein kleiner Teil der Datenpunkte zufällig gewählt und durch den Algorithmus verarbeitet wird. Durch diese Methodik der immer neu gewählten Batches lässt sich dem Erkennen lokaler Minima entgegenwirken und die Performanz steigt erheblich. Dies wird so oft durchgeführt, bis das Ergebnis der verschiedenen Durchführungen gegen ein Ergebnis konvergiert oder eine gegebene Anzahl an Iterationen durchgeführt wurde.

Content-based und syntactical features:

Im Allgemeinen kann man bei der Autorenklassifikation zwei Kategorien von Features unterscheiden, die unabhängig voneinander oder aber kombiniert untersucht werden können. Hierbei handelt es sich um inhaltsbasierte (content-based) und syntaktische Features. Inhaltsbasierte Features legen den Fokus auf lexikalische Merkmale, also den Inhalt eines Textes. Es wird vor allem auf die Semantik von Wörtern und Sätzen geachtet. Im Gegensatz dazu liegt der Fokus bei syntaktischen Features auf den Wörtern und Zeichen in einem Text, sowie ihre Relation zueinander. Syntaktische Features sind beispielsweise Subjekte, Objekte, Prädikate oder Attribute.

2.2 Related Work

Aufgrund des hohen Nutzens, den eine erfolgreiche Autorenklassifikation in den unterschiedlichsten Anwendungsbereichen stiften kann, gibt es eine umfassende Anzahl an wissenschaftlichen Veröffentlichungen mit Lösungsansätzen im Bereich Machine Learning.

Fatima et al. [6] erstellten zur Bewältigung dieser Thematik den *SMS-AP-18 Corpus*. Der Corpus enthält 810 Autorenprofile, wobei jedes Profil aus einer Aggregation von SMS-Nachrichten eines einzelnen Autors und sieben demographischen Merkmalen besteht. Die SMS-Nachrichten sind in Englisch und Roman Urdu verfasst. Der Datensatz wurde unter Anwendung von stilometrie- und inhaltsbasierten Methoden untersucht, um das Geschlecht der Autoren zu bestimmen. Die stilistischen Verfahren konzentrierten sich auf lexikalische Wörter, sowie die Anzahl von Absätzen und Verben. Ein lexikalisches Wort ist dadurch

definiert, dass es eine gemeinsprachliche Bedeutung hat, welche in allen grammatischen Varianten die gleiche ist. Das können Bezeichnungen für Eigenschaften, Gegenstände, Tätigkeiten, Umstände und Vorgänge sein. Die inhaltsbasierte Methode nutze N-Grams zur Identifikation von Alter und Geschlecht. N-Grams zerlegen Texte in Fragmente, wobei N aufeinanderfolgende Fragmente als N-Gramm zusammengefasst werden. Die Fragmente können Buchstaben, Wörter oder Ähnliches sein. Die Ergebnisse zeigen, dass die inhaltsbasierte Methode mit 5-N-Grams einen F1-Score von 0,947 und eine Accuracy von 0,975 erreichte und damit alle anderen Verfahren, die in dieser Studie betrachtet wurden, übertreffen konnte.

Devi et al. [19] haben ebenfalls eine Autorenklassifikation mit den Merkmalen Alter und Geschlecht durchgeführt und nutzten dafür 500 Autorenprofile aus dem *SMS-AP-18 Corpus*, wovon 150 zum Testen und 350 zum Trainieren der Modelle verwendet wurden. Das Merkmal Geschlecht hatte, wie alle Related Work, die Ausprägungen *männlich* oder *weiblich*. Das Alter wurde in drei Kategorien eingeteilt: 15-19 jährige, 20-24 jährige und 25-xx jährige. Inhaltsbasierte Methoden wie Bag of Words (BoW) und Term Frequency - Inverse Document Frequency (TF-IDF) wurden verwendet, um Features aus dem Datensatz zu extrahieren. SVM wurde für die Klassifizierung genutzt. Eine 10-fold cross-validation diente der Identifikation der besten Modelle. Die Performance der erstellten Modelle wurde anhand ihrer Accuracy gemessen. Das beste Modell hat für das Merkmal Alter eine Accuracy von 0,857 und für das Merkmal Geschlecht von 0,643. Die gemeinsame Accuracy liegt bei 0,536.

Azmi und Al-Ghadir [7] untersuchten das Geschlecht von arabischen Social Media Nutzern. Hierfür wurden Posts von den beliebtesten Social Media Seiten in Saudi Arabien extrahiert. Zunächst wurden die Daten im Preprocessing bereinigt. Dabei wurden alle Ziffern, Sonderzeichen und diakritischen Markierungen nicht-arabischer Wörter entfernt. Außerdem wurden alle Wörter normalisiert und tokenisiert. TF-IDF-Methoden wurden angewandt, um die Gewichtung einzelner Wörter in den Posts zu bestimmen. Anschließend wurde anhand dieser Gewichtung eine top- k Liste für Wörter und eine top- k Liste für Stämme erstellt. Top- k ist also ein Merkmalsvektor, der die Einträge mit der höchsten k -Rangfolge enthält. Zur Bestimmung des Geschlechts kamen SVM und KNN zum Einsatz. Zur Evaluierung der Performance wurde ein Subdatensatz von 1200 Einträgen genutzt. Eine 10-fold cross-validation wurde zur Identifikation der besten Modelle verwendet. Das KNN-Modell erreichte mit 0,9316 eine höhere Accuracy als das SVM-Modell mit 0,8733.

Modaresi et al. [10] fokussierten sich im Rahmen der PAN Author Profiling Challenge 2016 auf die genreübergreifende Alters- und Geschlechtsidentifikation. Als Genre kann eine bestimmte Ausprägung oder Klassifikation in der Literatur, der Kunst, der Musik, dem Film oder dem Journalismus bezeichnet werden. Beispiele für literarische Genres sind unter anderem *Fantasy*, *Horror*, *Krimi*, *Sachbuch*. Der bereitgestellte Trainingsdatensatz umfasste 463 Dokumente an Englischen Tweets, 250 Dokumente an Spanischen Tweets und 384 Dokumente an Niederländischen Tweets, wobei ein Dokument alle Tweets eines Autors

umfasste. Um die erstellten Modelle auf andere Datenquellen als Twitter zu evaluieren wurde der PAN2014 verwendet, welcher Posts aus anderen Social Media Plattformen enthält. Da die Datenquelle Genre der Trainings- und Testdaten nicht identisch ist, wurden die Trainingsdaten so bearbeitet, dass die meisten genrespezifischen Informationen eliminiert wurden. So konnte das Risiko von Overfitting auf andere Genres als Twitter reduziert werden. Für die Feature extraction wurden Word Unigrams, Word Bigrams, Character 4-Grams, der Average Spelling Error und Punctuation Features verwendet. Das finale Modell wurde mit einer Logistic Regression trainiert, da dieser Klassifikator einen vergleichsweise hohen Bias und eine geringe Varianz hat. Da es sich um eine multivariate Klassifikation handelte, wurde die one-vs-rest Methode verwendet. One-vs-rest ermöglicht dabei die Zerlegung einer Mehrklassen-Klassifikation in ein binäres Klassifikationsproblem pro Klasse. Modaresi et al. haben außerdem Gradient Boosting und Random Forests als Klassifikatoren angewandt. Die mit der Logistic Regression erzielten Ergebnisse waren diesen beiden Klassifikatoren überlegen. Zur Evaluation der Modelle wurde eine 10-fold cross-validation auf dem Twitterdatensatz durchgeführt. Die höchste Accuracy für Geschlecht wurde mit 0,7564 für englische Posts erreicht. Die höchste Accuracy für das Alter wurde mit 0.5179 für spanische Posts erreicht.

Studie	Algorithmus	Corpus	Accuracy
Fatima et al.	5-N-Grams	SMS-AP-18	Alter: - Geschlecht: 0,975
Devi et al.	SVM	SMS-AP-18 Test-Train-Split: 150 / 350	Alter: 0,857 Geschlecht: 0,643
Azmi und Al-Ghadir	Logistic Regression	Twitter-Daten PAN2014	Alter: 0,5179 Geschlecht: 0,7564
Moladesi et al.	SVM KNN	Social Media Posts aus Saudi Arabien	Alter: - Geschlecht SVM: 0,8733 Geschlecht KNN: 0,9316

Tabelle 2.1: Ergebnisse der Related Work

Im Allgemeinen kann festgehalten werden, dass das Vorgehen bei allen betrachteten Studien sehr ähnlich ist. Unabhängig von der Größe des Datensatzes, der Sprache und der zu bestimmenden Merkmale begannen alle betrachteten Vorgehen mit einem Pre-Processing der Daten. Im nächsten Schritt wurden unterschiedliche Arten des Feature Engineering betrieben, wobei die Nutzung von TF-IDF besonders häufig vorgekommen ist. Im Rahmen der Modellerstellung wurde der Datensatz in Trainings- und Testdaten unterteilt. Unterschiede sind bei den verwendeten Datensätzen aufgefallen. Beispielsweise unterscheiden sich SMS-Daten stark von Twitter-Daten, da eine SMS pro Buchstabe kostet. Es werden also im Allgemeinen eher kurze Worte und Sätze verwendet. Im Gegensatz dazu sind Twitter-Zeichen kostenlos und theoretisch unbegrenzt. Daher sind die verwendeten Worte uneingeschränkter und weniger homogen, als bei den SMS-Daten. Unterschiede sind außerdem bei den verwendeten Klassifikatoren ersichtlich. Genutzte Klassifikatoren waren SVM, Logistic Regression, KNN, Gradient Boosting und Random Forests.

3 Praktische Umsetzung

Die Vorgehensweise der praktischen Umsetzung umfasst das Feature Engineering und Data Cleaning, die Data Transformation, das Supervised Learning und Parametertuning, sowie die Modellevaluation und ist in Abbildung 3.1 dargestellt. Die einzelnen Schritte werden in diesem Kapitel detailliert erläutert.

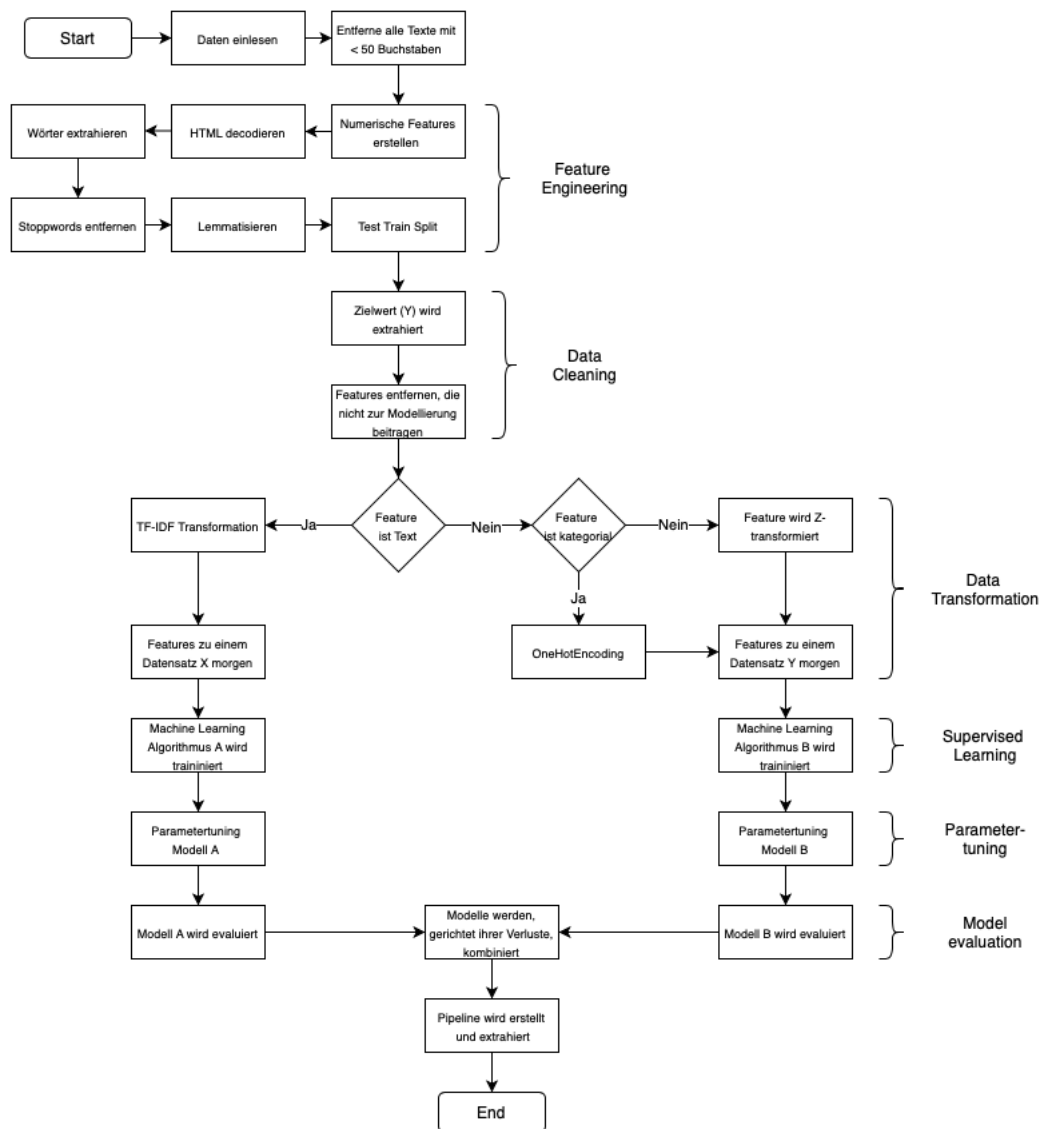


Abbildung 3.1: Visualisierung der Vorgehensweise
Quelle: Eigene Darstellung

Um die zuvor theoretisch erläuterten Konzepte im Folgenden praktisch zur Anwendung zu bringen, ist es zunächst notwendig, den Datensatz zu beleuchten und die darin enthaltenen Daten vorzuverarbeiten.

Datensatz:

Der in diesem Projekt vorwiegend verwendete *Blog Authorship Corpus* enthält 681.288 Blogeinträge aus dem Forum *blogger.com*. Über den reinen Textinhalt der Posts hinaus enthält er zudem Metadaten über 19.320 Verfasser. Diese Metadaten umfassen das Geschlecht, Alter und Sternzeichen, sowie das Thema und Veröffentlichungsdatum der Posts. Jeder Blogeintrag kann dabei eindeutig einem Autor zugeordnet werden. Dieser Datensatz wurde im Jahr 2004 zusammengetragen und gilt seither als renommierte Datenbasis für verschiedene Natural Language Processing (NLP)- und Klassifizierungsprojekte.

Noch vor der Vorverarbeitung des Datensatzes muss jedoch das Feature Engineering erfolgen. Hierbei wurden in Kollaboration die in Anlage A.1 beigefügten und erläuterten Features erarbeitet.

Zudem kann die Datenbasis durch ein Clustering angereichert werden, wodurch zusätzliche Informationen über den jeweiligen Autor entnommen werden können. Einerseits werden Cluster auf Basis der in Kapitel 3.1 beschriebenen textuellen Features erstellt. Ein weiteres Clustering basiert auf den stilometrischen Features. Die Interpretation der Cluster erfolgt manuell. Auf diese Weise entstehen für jeden dieser Blogeinträge zwei neue Features über den jeweiligen Autor, welche ebenfalls für neue Texte vorhergesagt werden können.

Frontend:

Eine grafische Benutzeroberfläche ermöglicht die leichte Verwendung der entstandenen Modelle. Von der Startseite aus kann jeder Benutzer zu einem Formular wechseln, in welchem er den Text wählt, dessen Autor in den nächsten Schritten analysiert werden soll. Die darunterliegende Dropdown-Liste ermöglicht die Wahl der vorherzusagenden Zielvariable. Dabei kann zwischen dem Alter, dem Geschlecht, dem Genre und dem Sternzeichen gewählt werden. Im folgenden Schritt müssen Werte für die übrigen drei Variablen angegeben werden, was voraussetzt, dass diese bereits bekannt sind. Anschließend werden die Ergebnisse der Klassifikation ausgegeben. Diese basieren auf dem gewichteten Stacking des textuellen und numerischen Modells. Auf der Ergebnisseite wird links oben der Eingabetext wiederholt. Rechts davon ist das Ergebnis des Genres und des Alters des Autors zu sehen. Unten links wird die Vorhersage für das Geschlecht, daneben das Ergebnis der aus dem Clustering gewonnenen Variablen und das Sternzeichen ausgegeben. Je nachdem welche Variable vorab als Unbekannte ausgewählt wurde, bezieht sich nur eines der Ergebnisse und das Clustering auf die berechnete Prognose. Die anderen basieren hingegen auf der vorangegangenen manuellen Eingabe. Unter dem Überblick zu den Eigenschaften wird ein maschinell generierter Text angezeigt, welcher dem Eingabetext ähnelt. Über die Menüleiste kann zwischen der Startseite, der Texteingabe, dem zuletzt erhaltenen Ergebnis und einer Übersicht der an dieser Arbeit beteiligten Teammitglieder gewechselt werden.

3.1 Vorverarbeitung

Für dieses Projekt ist eine individuelle Vorverarbeitung unabdinglich, um die Basis für die folgenden Machine Learning Schritte zu legen.

Cleaning:

- (1) Im Schritt des Cleaning werden dafür zum einen Verlinkungen wie z. B. „@User123“ aus den Blogeinträgen entfernt, um lediglich die Inhalte der verschiedenen Einträge eigenständig betrachten zu können.
- (2) Zum anderen wird auch die Zeichencodierung von der ursprünglich bestehenden HTML-Codierung zu einer UTF-8-Codierung abgeändert. UTF-8 ist eine der gängigsten Unicode Codierungen und daher für die folgende Verarbeitung optimal geeignet. Dadurch sind fort folgend auch z. B. keine chinesischen Schriftzeichen ohne Erweiterung abbildbar. Die HTML-Tags fallen in diesem Zuge ebenfalls weg.
- (3) Als nächstes werden sogenannte Stopwords wie z. B. „me“, die in einem Dokument vordefiniert sind und sich an den gängigen Standards orientieren. Auch Sonderzeichen wie z.B. „§“ werden aus dem Text entfernt, da diese in diesem Kontext nicht als relevant für die Informationsgewinnung erachtet werden. Sie kommen sehr oft und auch in verschiedensten Kontexten vor und dienen vor allem der grammatikalischen Richtigkeit, die für die folgenden Machine Learning Methoden nicht relevant ist. Sie würden die Verarbeitung daher vor allem verzerren, da sie den Großteil der Worte ausmachen, jedoch nicht den Hauptanteil an der Informationsgewinnung beisteuern. Zur Unterstützung dieser Textbereinigung wird die Bibliothek TextBlob <https://textblob.readthedocs.io/en/dev/> verwendet, um die erwünschten Worte zu erkennen und unerwünschten Inhalte zu entfernen.
- (4) Die Python Bibliothek spaCy <https://spacy.io/> unterstützt im letzten Schritt der Vorverarbeitung die Lemmatisierung der Wörter in den Blogeinträgen. Hierbei werden sie in ihre Grundform zurückgeführt und grundsätzlich klein geschrieben. So wird z.B. aus „Tischfußball“ „tisch“, „fuß“ und „ball“. Das dient der Verringerung der Heterogenität der Blogeinträge und steigert somit die Vergleichbarkeit. Im Verlauf wird diese Vereinfachung z.B. die Verwendung des Sprachfilters vereinfachen, der ebenfalls durch die Voreinstellung der Sprache Englisch in der Bibliothek FastText <https://fasttext.cc/> problemlos umgesetzt wird.

Transformation:

Die zuvor beschriebenen Schritte sind jedoch nicht ausreichend, um mit der Modellerstellung zu beginnen. Daher wird im Folgenden der Vorgang der Transformation beschrieben. Allgemein wird bei der Modellerstellung auf zwei grundsätzlich verschiedenen Datensätzen gearbeitet, um eine größere Diversität zu gewährleisten und dadurch im späteren Verlauf unterschiedliche Vorgehensweisen umzusetzen. Dies ist nötig, um sowohl numerische als

auch textuelle Features sachgerecht durch unterschiedliche Lernalgorithmen in die verschiedenen Modelle miteinzubeziehen. Bevor diese grundsätzlich verschiedenen Datensätze erstellt werden, werden noch die vorverarbeiteten Datensätze zufällig in je einen Trainingsdatensatz mit 481959 Einträgen und einen Testdatensatz mit 120490 Einträgen aufgeteilt. Danach kommen verschiedene sogenannte „Transformer“ zum Einsatz, um die heterogenen Daten nach numerischen bzw. textuellen Features der Metadaten zu trennen. Der textuelle Transformer ist definiert durch die Methode `TfidfVectorizer`, um die nominalskalierten Daten verarbeiten zu können. TF-IDF ist bestimmt durch die beiden Methoden TF und IDF und wird wie folgt definiert:

For a term i in a document j :

$$w_{i,j} = t_{i,j} * \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j

df_i = number of documents containing i

N = total number of documents

TF setzt dabei die Wortanzahl eines gewählten Wortes mit allen Wörtern im Blogeintrag ins Verhältnis und stellt somit die relative Wahrscheinlichkeit dar, dass ein zufälliges Wort mit dem zuvor gewählten Wort übereinstimmt. IDF setzt dagegen logarithmisch die Anzahl an Blogeinträgen mit der Gesamtanzahl der Blogeinträge, die ein bestimmtes Wort enthalten, ins Verhältnis und zeigt somit die Häufigkeit der Wörter blogübergreifend auf. TF-IDF wird durch die Multiplikation von TF und IDF berechnet und ist folglich ein Indikator für die Relevanz der Worte in den verschiedenen Blogeinträgen im gesamten Datensatz. Diese gesamten Informationen werden in einem Vektor gespeichert. Durch das folgende Beispiel soll diese Methodik genauer erläutert werden:


Es wird die Annahme getroffen, dass in einem Dokument mit 100 Worten drei mal das Wort „Kartoffel“ vorkommt. Daraus folgt gemäß der Formel der $TF - Wert = \frac{3}{100} = 0,03$. Dieses eine Dokument wird in diesem Beispiel als eines von 10.000.000 betrachtet, in denen das Wort „Kartoffel“ 1.000 mal vorkommt. Nach der Formel ergibt sich für $IDF = \log \frac{10.000.000}{1.000} = 4$. Daraus folgt der $TF - IDF = 0,03 * 4 = 0,12$. Diese Berechnung wird für jedes der relevanten Worte aller Dokumente durchgeführt und als Wert in einem Vektor

$$TF - IDF = \begin{pmatrix} 0,12 \\ \dots \\ \dots \end{pmatrix}$$

gespeichert.

Dieser Transformator ist aufgrund seiner umfassenden Betrachtung der gesamten Kollektion der Blogeinträge und jedes einzelnen Blogeintrags in Bezug auf die verwendeten Wörter optimal für den Anwendungsfall geeignet. Andere Methoden wie z.B. ein Count Vectorizer betrachtet einzig und allein die Anzahl der verschiedenen Wörter in einem einzelnen Datensatz. Dadurch wäre es entweder nicht möglich ohne weitere Methoden die verschiedenen Blogeinträge übergreifend zu bewerten oder nach den einzelnen Einträgen zu differenzieren. Angesichts des wissenschaftlichen Anspruchs dieser Arbeit wurde jedoch auch in Teilen der IDF-Faktor des TF-IDF Transformators ausgeschlossen, um nachzuvollziehen, ob dieser überhaupt einen nützlichen Einfluss auf das Ergebnis hat. Darauf wird in Kapitel 4, der Bewertung weiter eingegangen. Die Methode TF-IDF wird vollumfänglich auf den bis hierher vorverarbeiteten Text angewandt.

Der numerische Transformator bedient sich bei kategorialen Features des OneHotEncoders, bei z.B. fortlaufenden numerischen Ausprägungen des Standardscalars. Diese Unterscheidung ist nötig, da kategoriale Features entweder ordinal- oder kardinalskaliert sein können und dadurch eine Unterscheidung in der weiteren Verarbeitung unabdinglich ist.



Color		Red	Yellow	Green
Red		1	0	0
Red		1	0	0
Yellow		0	1	0
Green		0	0	1
Yellow		0	1	0

Abbildung 3.2: One Hot Encoder
Quelle: In Anlehnung an [4]

One Hot Encoder, wie in Abbildung 3.2 beispielhaft zu sehen, ordnen die verschiedenen ordinalskalierten Ausprägungen des jeweiligen Features ihren entsprechenden Kategorien zu. Diese Methode wird ausschließlich für die Zielvariablen „Thema“, „gender“ und „sign“ verwendet.

Der Standardscalar hingegen standardisiert die verschiedenen kardinalskalierten Ausprägungen wie bei einer z-Transformation, sodass sich der Mittelwert bei 0 befindet und die Standardabweichung 1 beträgt. Dadurch ergibt sich eine bessere Vergleichbarkeit und eine erleichterte Bewertbarkeit der Ausprägungen, da sich ca. 68 % im Bereich von -1 bis 1 befinden. Die z-Transformation wird einerseits für die erstellten Features, andererseits aber auch auf die Zielvariable „age“ verwendet.

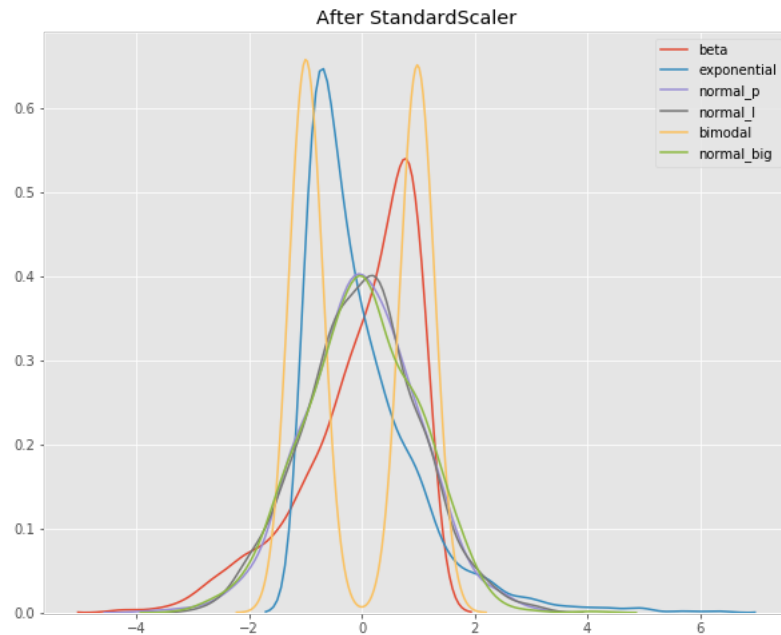


Abbildung 3.3: StandardScaler
Quelle: Entnommen aus [8]

Diese Transformer werden durch die Sklearn Methode `make.column.transformer` zu Tupeln mit dem Inhalt (Transformer, Datenanteil) ihrem Datenanteil zugewiesen und auf diesen angewendet. So erhalten wir Datensätze getrennt nach numerischen oder textuellen Features. Diese Teildatensätze X bzw. Y können nun auf die verschiedenen Arten der Features passend weiterverarbeitet und in Modellen berücksichtigt werden. Tabelle 3.1 zeigt die Zuordnung von Transformer und Variablen dieser Ausarbeitung.

Transformator	Variablen
TF-IDF	gesamter vorverarbeiteter Text
OneHotEncoding	Topic, Gender, Sign
Z-Transformation	alle erstellten Features, Age

Tabelle 3.1: Zuordnung der Transformatoren zu den verarbeiteten Variablen

3.2 Linear Support Vector Machine

Im Folgenden wird erklärt, wie die verschiedenen Modelle erstellt werden. Dabei wird zunächst die Methode der LSVM und ihre Anwendung auf textuelle und numerische Features erklärt. Grundsätzlich ist es wie auch bei der klassischen linearen Regression das Ziel der

LSVM, die gegebenen Datenpunkte durch eine lineare Funktion darzustellen. Um die Datenpunkte durch eine Funktion zu approximieren, werden jedoch nicht die einzelnen Werte des ordinary least squares der Datenpunkte minimiert, sondern die Werte der L2-Norm des Koeffizientenvektors der Datenpunkte. Da der Error nicht direkt zur Erstellung der Funktion verwendet wird, kann er als vorausgesetzter Wert angegeben werden, um die Accuracy des Modells zu verbessern. Somit kann man durch den angegebenen Error die Datenpunkte mit einer gewissen Toleranz „c“ betrachten. Das führt zu einer Modellverbesserung, da die Datenpunkte durch den durch „c“ beschriebenen Bereich besser beschrieben werden als durch eine einfache lineare Funktion. Durch verschiedene Hyperparameter wie z.B. Schlupfvariablen können zusätzlich die Datenpunkte miteinbezogen werden, die nicht in den Toleranzbereich um die Funktion fallen. Schlupfvariablen erfassen die Informationen dieser Datenpunkte durch ihre Abstände vom tolerierten Bereich bzw. der nächsten Grenze dieses. Zusätzlich zur L2-Norm sollen auch diese Schlupfvariablen minimiert werden. Zudem wird auch der zuvor beschriebene angegebene Error bzw. die daraus entstehenden beiden Toleranzgrenzen „c“ als Hyperparameter betrachtet. Dieser und die Schlupfvariable werden in Abhängigkeit zu einander folglich optimiert, sodass der Bereich um die lineare Funktion die Datenpunkte bestmöglich abdeckt, ohne zu ungenau zu werden. Wird „c“ zu groß gewählt, ist dieser sehr groß und der Wert der Schlupfvariable sehr klein und es kommt zum underfitting. Ist „c“ sehr klein gewählt, so ist der Wert der Schlupfvariable sehr groß und es kommt zu Overfitting. Dazwischen muss ein passender Kompromiss gefunden werden, um die Vorhersage des Modells zu optimieren. Diese Optimierung kann z.B. durch eine Gridsearch durchgeführt werden. Somit ist eine LSVM eine komplexere Erweiterung der klassischen Regression bzw. Klassifikation durch Toleranzen und einen festgesetzten Error und ist dadurch gut als Algorithmus für dieses Projekt geeignet.

SVM können entweder als Regressor dienen, indem sie für numerische Daten das oben beschriebene Modell als Funktion für den Output eines fortlaufenden Wertes berechnen, oder als Klassifikator. Als Klassifikator dient die Funktion als „Decision Boundary“ zwischen den verschiedenen Klassen. Bei einer binären Klassifikation wird dabei nur eine Decision Boundary mit Toleranzbereich berechnet, bei einer multiclass Klassifikation dementsprechend mehrere.

Je ein optimiertes Modell wird für jedes der numerischen und textuellen Features auf dieser Grundlage trainiert und getestet. Dies dient im Folgenden dem Vergleich mit den beiden anderen Modellarten, die durch andere Methoden trainiert werden.

Tabelle 3.2 zeigt die optimale Hyperparameterkombination für jede Zielvariable. Neben den optimalen Ausprägungen der Parameter wurden außerdem noch weitere in jeder möglichen Kombination für das numerische und das textuelle Modell untersucht:

- (1) Berechnung des Parameter Losses entweder durch „epsilon insensitive“, „squared epsilon insensitive“, „hinge“ oder „squared hinge“.
- (2) Für den Parameter c wurden die Werte 0,8 oder 1 oder 1,2 verwendet.

Zielvariable	Kategorie	Parameter	Optimaler Wert	Metrik
Alter	numerical	Loss = squared_epsilon_insensitive C = 0,8	-46,1547	MSE
Alter	text	Loss = squared_epsilon_insensitive C = 0,8	-41,8165	MSE
Geschlecht	numerical	Loss = Squared Hinge C = 1,2 Penalty = L2	0,618599	Accuracy
Geschlecht	text	Loss = Hinge C = 0,8 Penalty = L2	0,679198	Accuracy
Sternzeichen	numercial	Loss = Squared Hinge C = 1,2 Penalty = L2	0,129823	F1-Score
Sternzeichen	text	Loss = Squared Hinge C = 0,8 Penalty = L2	0,163772	F1-Score
Thema	numerical	Loss = Squared Hinge C = 1 Penalty = L2	0,320926	F1-Score
Thema	text	Loss = Squared Hinge C = 0,8 Penalty = L2	0,331889	F1-Score

Tabelle 3.2: Optimale Parameter für die LSVM

3.3 Logistische Regression

Grundsätzlich geht es auch bei der logistischen Regression darum, die gegebenen Daten in einem Modell bzw. einer Funktion darzustellen, um neue Datenpunkte darauf folgend z.B. klassifizieren oder konkrete Werte für diese vorhersagen zu können. Dabei können je nach Art der Regression eine oder mehrere Einflussvariablen berücksichtigt werden. Welche Struktur die Grundfunktion hat, hängt davon ab, wie sich die Verteilung der Datenpunkte gestaltet. So ist es möglich, dass es eine lineare Funktion ist. In unserem Anwendungsfall ist die Funktion jedoch durch die folgende s-förmige Funktion beschrieben:

$$L(y_i, f(x_i)) = \log(1 + \exp(-y_i * f(x_i)))$$

Die Besonderheit der logistischen Regression ist, dass sie sich zwischen den Funktionswerten 0 und 1 befindet. So geht sie bei Inputwerten, die gegen 0 gehen auch gegen 0, bei Inputwerten, die gegen ∞ gehen, geht sie in Richtung 1. Somit wird sie als Wahrscheinlichkeitsverteilung betrachtet und dienen vornehmlich dem Zweck der Klassifikation. Die Outputwerte entsprechen dabei der Wahrscheinlichkeit, dass der Inputwert zur Klasse 0 bzw. 1 zugeordnet wird. Somit ist in diesem Fall die Funktion keine decision Boundary, sondern ordnet vielmehr den Inputwert zu einer gewissen Wahrscheinlichkeit der Klasse 1 zu. Die Wahrscheinlichkeit der Zuordnung zu Klasse 0 lässt sich durch 1-Outputwert

bestimmen.

Es wird jedoch nicht nur eine Funktion approximiert, sondern viele verschiedene, bis die passendste all dieser Funktionen durch den SGD auf Grundlage der maximum likelihood gefunden werden kann. Für die durch Klassifizierung gelösten Zielvariablen Geschlecht, Sternzeichen und Thema werden auch durch diese Methodik verschiedene Modelle gelernt, die in 4 mit einander in Konkurrenz stehen.

Die optimale Parameterkonstellation für die Zielvariablen sind in Tabelle 3.3 aufgeführt.

Zielvariable	Kategorie	Parameter	Optimaler Wert	Metrik
Geschlecht	numerical	alpha = 0,001 Loss = log Penalty = L2	0,617497	Accuracy
Geschlecht	text	alpha = 0,00001 Loss = log Penalty = elastic net (L1+L2)	0,683297	Accuracy
Sternzeichen	numerical	alpha = 0,001 Loss = log Penalty = L2	0,126876	F1-Score
Sternzeichen	text	alpha = 0,00001 Loss = log Penalty = elastic net (L1+L2)	0,165655	F1-Score
Thema	numerical	alpha = 0,001 Loss = log Penalty = elastic net (L1+L2)	0,321668	F1-Score
Thema	text	alpha = 0,00001 Loss = log Penalty = L1	0,324179	F1-Score

Tabelle 3.3: Optimale Parameter für die logistische Regression

Auch bei dieser Methodik wurden verschiedene Parameterkonstellationen geprüft, dabei waren sowohl für die textuellen, als auch für die numerischen Modelle bei den folgenden Parametern folgende Ausprägungen möglich:

- (1) Berechnung der Parameter Penalty entweder durch die L2-Norm, durch die L1-Norm oder durch eine Kombination dessen („elasticnet“).
- (2) Für den Parameter α wurden die Werte 0,001 oder 0,00001 verwendet.

3.4 Extreme Gradient Boosting

XGBoost ist eine erweiterte Version des Gradient Boosting. Das Gradient Boosting erzeugt grundsätzlich eine große Anzahl verschiedener „einfacher“ Modelle auf kleinen Anteilen des Datensets und kombiniert diese wie in einem Regressiiontree bzw. Decisiontree. Diese Modelle werden dann nach ihren Vorhersagen und ihrem Loss gewichtet. Die schlechter

abschneidenden Modelle werden in den folgenden Schritten intensiver trainiert als jene, die von Anfang an gute Vorhersagen treffen. Über den Gradienten wird auch hier wie im Gradient Descent die Lossfunction der verschiedenen Modelle möglichst minimiert. Schlussendlich werden die verschiedenen Modelle kombiniert, um ein bestmögliches Ergebnis bei der Vorhersage zu erzielen.

XGBoost ist eine spezielle Umsetzung dieses Ansatzes, die statt des Gradienten Verfahrens das Newton Verfahren verwendet und dadurch über mehr Informationen wie z.B. die Annäherung an die Lossfunction über zweite Ableitung verfügt und dadurch effektiver das Minimum findet. Zudem verwendet XGBoost die Regulierungsterme L1 und L2. Das führt zu einer besseren Verallgemeinerung des finalen Modells, da nicht nur der Error betrachtet wird.

Hier kann je nach Feature über den Regressionstree bzw. Decisiontree eine Vorhersage über den Wert bzw. die Klasse getroffen werden. Je ein Regressionstree wird für alle numerischen Features und je ein Decisiontree wird für allem zu klassifizierenden features als Modell erstellt, um wiederum ein weiteres Modell zur Auswahl vergleichen zu können.

Zielvariable	Kategorie	Parameter	Optimaler Wert	Metrik
Alter	numerical	learning rate = 1 max depth = 6 n estimators = 400	-44,75678	MSE
Alter	text	learning rate = 1 max depth = 6 n estimators = 400	-55,0873	MSE
Geschlecht	numerical	learning rate = 0,1 max depth = 6 n estimators = 400	0,72342	Accuracy
Geschlecht	text	learning rate = 0,1 max depth = 6 n estimators = 800	0,66244	Accuracy
Sternzeichen	numerical	learning rate = 0,1 max depth = 6 n estimators = 800	0,353057	F1-Score
Sternzeichen	text	learning rate = 0,1 max depth = 6 n estimators = 400	0,157878	F1-Score
Thema	numerical	learning rate = 0,1 max depth = 6 n estimators = 400	0,473595	F1-Score
Thema	text	learning rate = 0,1 max depth = 6 n estimators = 400	0,315916	F1-Score

Tabelle 3.4: Optimale Parameter für den XGBoost

Auch durch diese Methode sind im Prozess des Hyperparametertunings einige Modelle entstanden. Das optimale Modell für jede Zielvariable und die dazugehörigen Hyperparameterwerte sind in Tabelle 3.4 zu erkennen.

Über die Parameterkonstellation hinaus, die sich als optimal herausstellte, wurden sowohl für die numerischen, als auch die textuellen Modelle auch die folgenden Parameterausprägungen mit einander beliebig kombiniert:

- (1) Für den Parameter „learning rate“ wurden hierbei die Werte 0,1 oder 1 oder 1,5 verwendet.
- (2) Für den Parameter „max depth“ wurden hier die Werte 3 oder 6 verwendet.
- (3) Bei dem Parameter „n-estimators“ wurden entweder 400 oder 800 estimators verwendet.

3.5 Clustering mittels K-means

K-means ist ein Algorithmus, welcher für das Clustering verwendet werden kann, wodurch neue Features im Datensatz sichtbar gemacht werden. Dabei müssen die einzelnen Datenpunkte unterschiedlichen Clustern zugeordnet werden, welche in sich homogen und nach außen hin heterogen sind.[3] Jedes Cluster i wird dabei durch einen Clustermittelpunkt μ_i repräsentiert. Die Zuordnung erfolgt mit dem Ziel, die Summe der quadrierten Distanzen J zwischen den einzelnen Datenpunkten und Clustermittelpunkten, welche für den jeweiligen Datenpunkt gewählt wurde, zu minimieren. Um dies zu erreichen, wird jeder Datenpunkt natürlich immer dem ihm am nächsten gelegenen Clustermittelpunkt μ_i zugeordnet.[3] Um nun das Fehlermaß J weiter zu verringern, setzt ein iterativer Prozess ein, welcher den Schritten des EM-Algorithmus gleicht.

Zu Beginn muss die Anzahl der Cluster k definiert werden. Hierbei handelt es sich um einen in einer Validierung zu optimierenden Hyperparameter, wobei der sich ergebende Ellbogenpunkt als k gewählt wird. Anschließend werden die Mittelpunkte aller Cluster erstmals zufällig definiert und alle Datenpunkte dem jeweils nächst gelegenen Cluster zugeordnet. Der folgende Schritt aktualisiert die Clustermittelpunkte, in dem der Mittelwert der soeben zugeordneten Datenpunkte berechnet und als neuer Mittelpunkt festgelegt wird. Den neu gewonnen Mittelpunkten können die Datenpunkte nun wiederholt zugeordnet und die Mittelpunkte erneut aktualisiert werden. Dieser iterative Prozess terminiert, sobald die Aktualisierungen konvergieren.[3]

In der Praxis kann ein K-means Modell mittels der Python-Bibliothek scikit-learn umgesetzt werden. Dabei lässt sich die Anzahl der Iterationen alternativ beispielsweise auf 200 begrenzt werden, sodass eine Termination zugunsten einer gewissen Zeitersparnis erzwungen wird.

4 Diskussion der Ergebnisse

Allgemein lässt sich festhalten, dass dieses Projekt vom grundsätzlichen Vorgehen her der Related Work 2.2 sehr ähnelt. Vor allem das Preprocessing und das Data Cleaning als Grundlage für die folgenden Modellberechnungen sind sehr ähnlich. Ein großer Unterschied liegt in der Menge der Daten. So wurde in vergleichbaren Projekten eine wesentlich kleinere Datenbasis verwendet. Jedoch führt ein größerer und diverserer Trainingsdatensatz meist zu besseren Modellen. Ein Overfitting oder Underfitting kann verhindert werden, da dem Modell ausreichende und sehr heterogene Daten zum Training bereitstehen. Es kann somit eine bessere Qualität der Vorhersagen erzeugt werden.

Die verwendete Datenquelle hat jedoch auch einige Nachteile, denn sie ist veraltet und kann daher den sprachlichen Wandel der vergangenen 15 Jahre nicht berücksichtigen. So kann es sein, dass die Verwendung verschiedener Textcharakteristika eher einem Geburtszeitraum als einem pauschalen Alter zugeordnet werden können. Möglicherweise „altern“ die Gewohnheiten mit den Verfassern. So ist es möglich, dass Charakteristika, die dem Modell nach etwa 15-Jährigen zuzuordnen werden, nun eher in der Verwendung von etwa 30-Jährigen sind. Diese Möglichkeit wird in der hier ausgeführten Arbeit nicht näher beleuchtet, da der Test ebenfalls mit Daten des gleichen Zeitraums umgesetzt wird.

Ein weiterer Nachteil ist, dass nur Blogbeiträge von 13- bis 47-Jährigen im Datensatz enthalten sind und somit eine Vorhersage für jüngere oder ältere Personen kaum möglich ist. Zudem wird das Ergebnis dieses Projekts nicht problemlos Merkmale aller Texttypen vorhersagen können, da die Modelle lediglich auf Blogbeiträgen erstellt wurden. Blogbeiträge haben in vielerlei Hinsichten grundsätzlich andere Charakteristika wie unter anderem eine informellere Sprache im Vergleich zu Sachbüchern. Auch haben sie eine höhere Informationsdichte als Kinderbücher und sind vor allem deutlich kürzer als diese, um nur einige Beispiele zu nennen. Daraus folgt, dass das Ergebnis dieses Projekts nicht für jeden Texttyp, jedoch gut für Blogbeiträge geeignet ist.

Abschließend ist auch zu bemerken, dass sowohl das Feature Engineering als auch die verschiedenen Methodiken zum Erstellen und Validieren der Modelle deutlich ausgeprägter stattgefunden haben, als in vergleichbaren Projekten. Das Feature Engineering wurde stärker beleuchtet, wodurch sowohl Klassifizierung, Regression und Clustering Anwendung auf den Datensatz finden. Bei jeder dieser Vorgehensweisen wurden verschiedene Methoden verwendet und die Ergebnisse gegeneinander abgewogen, statt die Umsetzung einer Methodik explizit zu demonstrieren.

4.1 Bewertung der Modelle für die Zielvariable Alter

Bei der Messung der Zielvariable Alter mittels negativem Mean Squared Error (MSE) sind einige interessante Effekte zu erkennen. Zunächst wird auf die textuellen Modelle eingegangen. Hier ist zu erkennen, dass die Modelle der drei Methoden XGBoost des Kapitels 3.4, logistische Regression des Kapitels 3.3 und LSVM des Kapitels 3.2 im Vergleich zu den anderen Zielvariablen ungleiche Ergebnisse erzielt haben. Der zum Vergleich herangezogene minimale MSE liegt zwischen -40 und etwa -55. Dabei liegt das LSVM-Modell bei einem MSE von -41,82, das Modell der logistischen Regression liegt bei -54,23 und das XGBoost-Modell liegt bei -55,09. Auf die Qualität der Modelle haben je die Hyperparameter der TF-IDF oder wahlweise nur der TF in Kombination mit der Range der N-Grams einen begrenzten Einfluss.

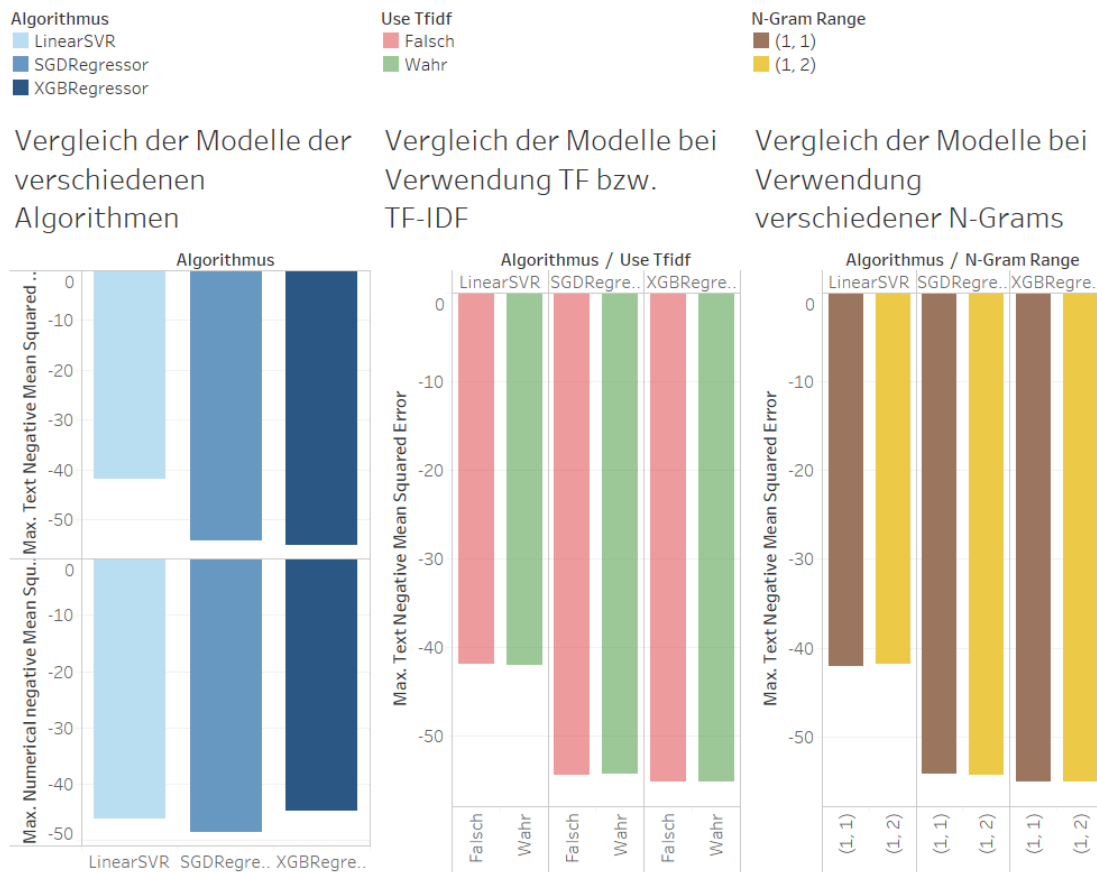


Abbildung 4.1: Bewertung der Modelle für die Zielvariable Alter

Quelle: Eigene Darstellung

So liegt in der Verwendung der TF-IDF bei den Modellen der logistischen Regression und

des XGBoost ein minimaler Vorteil gegenüber der Verwendung von lediglich der TF. Bei den LSVM-Modellen liegt jedoch ein Vorteil in der Verwendung der TF. Somit wurde sich bei den Modellen der logistischen Regression und des XGBoost für eine Verwendung der TF-IDF und bei den LSVM-Modellen nur für eine Verwendung der TF entschieden. Dadurch wird die Modellqualität je nach Bedürfnis der verschiedenen Algorithmen durch die Verwendung der TF bzw. der TF-IDF verbessert.

Eine weitere Einflussgröße ist die Größe der Range der N-Grams. Dabei wird ein Vergleich der Range von $+ - 1$ für jedes Wort mit der Betrachtung nur des Wortes an sich angestrebt. Bei den Modellen der Algorithmen XGBoost und LSVM ist dabei eine Verbesserung durch die Erweiterung der N-Grams zu erkennen. Bei den Modellen der logistischen Regression jedoch wird keine Verbesserung deutlich. Somit wird auch in der Modellerstellung mittels logistischer Regression die einfachere Methode der N-Grams (1,1) verwendet. Da bei der Modellerstellung mittels XGBoost und der LSVM eine minimale Verbesserung zu verzeichnen ist, wird hier die komplexere Variante der N-Grams (1,2) zur Modellerstellung verwendet.

Bei den optimalen numerischen Modellen der drei Methoden ist hier kein derart deutlicher Qualitätsunterschied zu erkennen. Die optimale Konfiguration der Hyperparameter zur Erstellung dieser Modelle ist in Abbildung *[wird ergänzt]* in Kapitel 3 beschrieben. Auch mit optimaler Konstellation der Hyperparameter schneidet das XGBoost-Modell mit einem MSE von deutlich über -45 etwas besser als die beiden Modelle ab, die durch die logistische Regression und die LSVM erstellt wurden. Diese beiden schlechteren Modelle erzielen nahezu die selbe Güte. Die Güte, die durch die LSVM erzielt wurde, ist jedoch etwa 2 höher, wodurch das Modell der logistischen Regression hier das schlechteste Ergebnis liefert. Allgemein ist zudem zu erkennen, dass die numerischen Modelle keinen großen Qualitätsunterschied gegenüber den textuellen Modellen aufweisen.

Abschließend wird im Frontend eine gewichtete Lösung des besten textuellen und des besten numerischen Modells bereitgestellt, um durch die Kombination das bestmögliche Ergebnis erzielen zu können. Die beiden besten Modelle werden nicht wie zuvor zur Bewertung der Algorithmen bzw. der dadurch erstellten Modelle mittels Crossvalidation trainiert, sondern nun zur Ausschöpfung aller Mittel auf dem gesamten Trainingsdatensatz.

Dadurch minimiert sich schlussendlich in der Anwendung der MSE beim textuellen Modell auf -4.648, der MSE des numerischen Modells auf -4.485 und der kombinierte gewichtete MSE auf -4.046. Somit wird für das Feature Alter das textuelle Modell, das durch den LSVM Algorithmus mit den optimalen Hyperparametern erstellt wurde und das numerische Modell, das durch den XGBoost mit den bekannten optimalen Parametern erstellt wurde gewichtet und eingebunden. Das textuelle LSVM-Modell erhält dabei eine Gewichtung von 49.109% und das numerische XGBoost-Modell von 50.891%.

4.2 Bewertung der Modelle für die Zielvariable Geschlecht

Bei den Modellen für die Zielvariable Geschlecht sind einige, jedoch von der Zielvariable Alter abweichende Effekte zu sehen. Bei dieser Zielvariable wird die maximale Accuracy zum Vergleich herangezogen, da der MSE nicht für die Klassifikation anwendbar ist. Sie verbessert im Gegensatz zum MSE mit steigendem Wert.

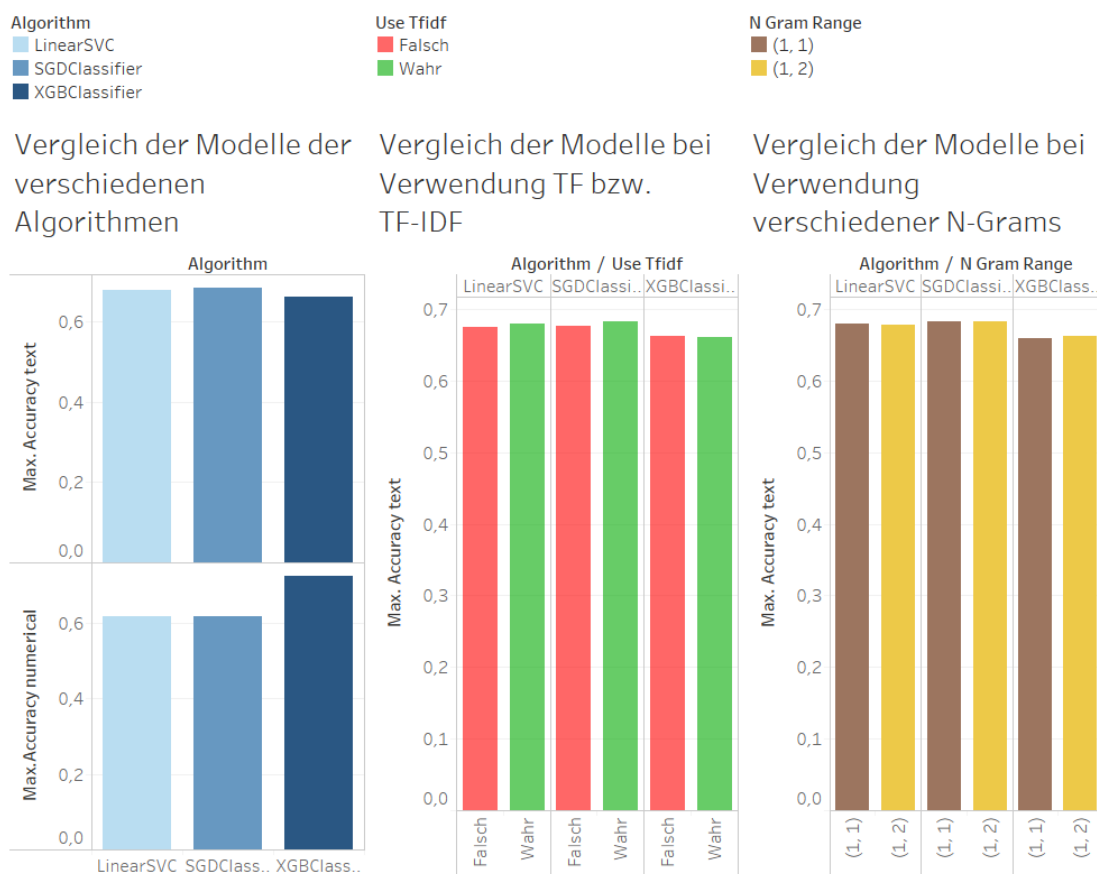


Abbildung 4.2: Bewertung der Modelle für die Zielvariable Geschlecht
Quelle: Eigene Darstellung

Die Zielvariable Geschlecht erreichte bei den textuellen Modellen eine Accuracy von etwa 70%. Der genaue Wert für das LSVM-Modell liegt bei 67,920%, für das Modell der logistischen Regression bei 68,330% und für das XGBoost-Modell bei 66,244%. Bei dieser Zielvariable hat die Verwendung des TF-IDF einen positiven Effekt auf die LSVM-Modelle und die Modelle der logistischen Regression. Beim XGBoost-Modell lässt sich dagegen keine Verbesserung erkennen. Somit werden für die Erstellung der LSVM-Modelle und der

Modelle der logistischen Regression TF-IDF und für die Erstellung der XGBoost-Modelle ausschließlich TF verwendet.

Bei den N-Grams ist algorithmusübergreifend nur ein marginaler Unterschied zu erkennen. Es konnte keine Verbesserung bei den Modellen der logistischen Regression und den LSVM-Modellen durch eine Erhöhung der N-Grams auf (1,2) belegt werden. Aufgrund dessen werden die N-Grams bei (1,1) belassen. Für die XGBoost-Modelle ist eine kleine Verbesserung zu erkennen und somit werden die N-Grams mit den Werten (1,2) für die optimale Modellerstellung verwendet.

Die Qualität der textuellen und numerischen Modelle ist vergleichbar. Bei den numerischen Modellen ist im Gegensatz zu den textuellen erneut ein deutlicher Qualitätsunterschied zu erkennen. Dabei erreicht das optimale XGBoost-Modell durch die in Kapitel 3 beschriebenen Hyperparameter eine Accuracy von 72,34%. Die beiden anderen Modellarten befinden sich dagegen bei einem Wert von rund 61%. Hier wird ebenfalls das beste textuelle Modell und das beste numerische Modell auf dem gesamten Trainingsdatensatz mit den bekannten Hyperparametern trainiert, um die Qualität zu verbessern. Dabei kommt die Kombination des textuellen und numerischen Modells auf eine Accuracy von 80,3%, obwohl die beiden einzelnen besten Modelle auch hier nur eine Accuracy von 70,23% beim textuellen Modell und eine Accuracy von 77,23% beim numerischen Modell erreicht haben. Auch für die Zielvariable Geschlecht wird eine gewichtete Kombination des jeweils besten Modells im Frontend für die Klassifizierung bereitgestellt. Diese kombiniert das numerische XGBoost-Modell zu 54.2661% und das textuelle Modell der logistischen Regression zu 45.7339%.

4.3 Bewertung der Modelle für die Zielvariable Sternzeichen

Bei der Zielvariable Sternzeichen wird die Bewertung der Ergebnisse anhand des F1-Scores getätigt. Der F1-Score ist ebenso wie die Accuracy zur Verbesserung zu maximieren. Daher ist auch hier zu erkennen, dass die textuellen Modelle nicht weit auseinander liegen. Das LSVM-Modell erreichte einen Wert von 16,3772%, das Modell der logistischen Regression von 16,5655% und das XGBoost-Modell von 15,7878%. Allgemein ist zu bemerken, dass ein F1-Score von 16% ein verhältnismäßig schlechtes Ergebnis darstellt. Die Ursache dessen könnte sein, dass das Sternzeichen keinen nachweisbaren Einfluss auf die Charaktereigenschaften eines Menschen und somit vermutlich auch keinen Einfluss auf den Schreibstil hat. Dennoch ist es eine interessante Zielvariable und kann im Rahmen dieser Möglichkeiten trotzdem im Frontend vorhergesagt werden.

Hier hat die Verwendung von TF bzw. TF-IDF bei den LSVM-Modellen und den XGBoost-Modellen kaum einen Einfluss. Trotzdem wird die bessere Einstellung dieses Hyperpara-

meters genutzt, was bei den XGBoost-Modellen die Nutzung von TF und bei den LSVM-Modellen die Nutzung von TF-IDF ist. Bei den Modellen nach der logistischen Regression ist hier jedoch eine etwas deutlichere Verbesserung durch die Verwendung von TF-IDF gegenüber TF zu erkennen und daher wird auch TF-IDF verwendet.

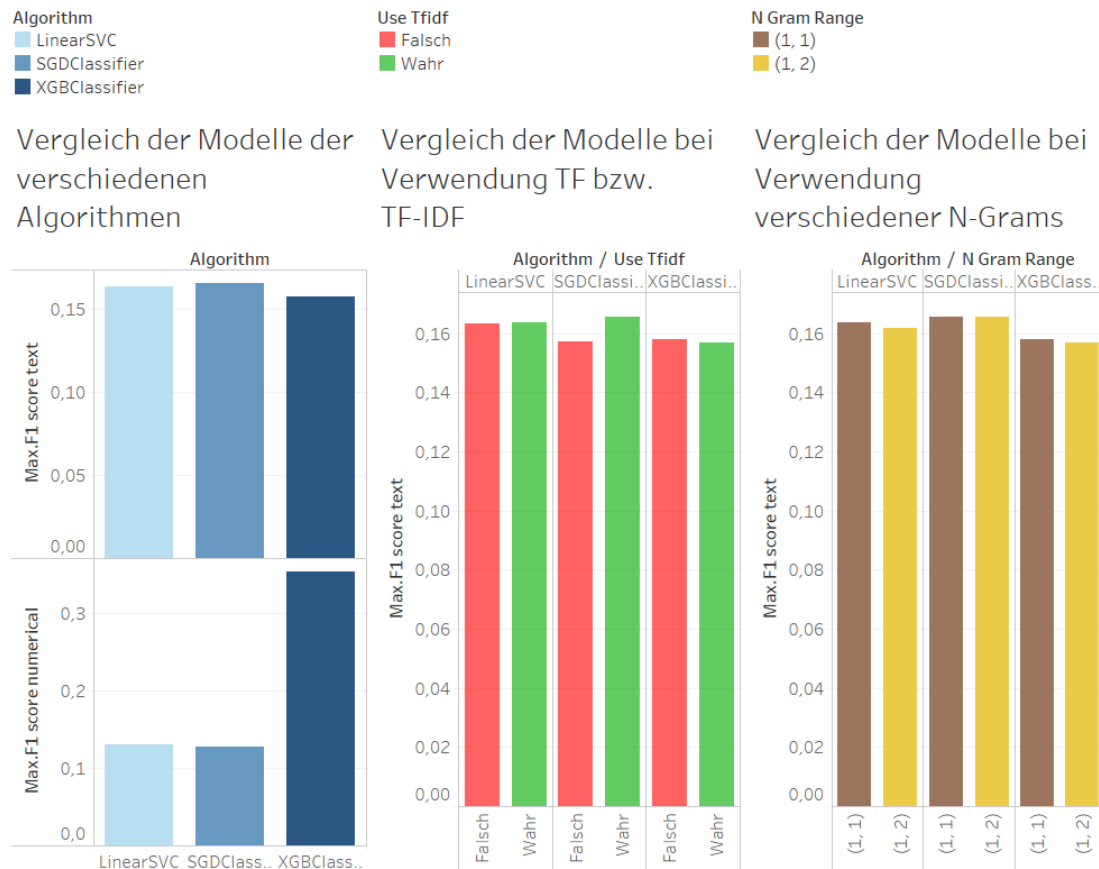


Abbildung 4.3: Bewertung der Modelle für die Zielvariable Sternzeichen
 Quelle: Eigene Darstellung

Gegensätzlich zu der Verwendung von TF bzw. TF-IDF sind bei der Verwendung der beiden Wertpaare (1,1) bzw. (1,2) der N-Grams erneut keine deutlichen Qualitätsunterschiede zu erkennen. Die Werte (1,1) führen bei den Modellen, die mittels der Algorithmen LSVM und XGBoost erstellt wurden, zu einer minimalen Verbesserung im Bereich der dritten Nachkommastelle des F1-Scores. Das Wertpaar (1,2) verbessert hier nur die Modelle nach der logistischen Regression. Diese Hyperparameter werden trotz ihres marginalen Einflusses auf die beschriebene Weise zugeordnet.

Wie bei den vorherig beschriebenen Zielvariablen ist auch bei der Zielvariable Sternzeichen eine Diskrepanz in der Qualität der verschiedenen numerischen Modelle erkennen.

Die LSVM-Modelle und die Modelle der logistische Regression erreichen einen Wert von knapp über 10%. Das XGBoost-Modell schneidet hier deutlich besser ab und erreicht einen Wert von 35,31%. Dieser ist über alle numerischen und textuellen Modelle übergreifend mit Abstand am besten.

Mit der ungefähren Verdopplung des F1-Scores bei dieser Variante ist für die Zielvariable Sternzeichen auch ein deutlicher Unterschied über die numerischen Modelle hinaus im Vergleich mit den textuellen Modellen zu erkennen. Auch für die Zielvariable Sternzeichen werden wieder je das beste textuelle und numerische Modell vollumfänglich auf dem Trainingsdatensatz trainiert und kommen somit auf einen F1-Score von 19,62% beim textuellen und 39,96% beim numerischen Modell. Trotzdem liefert das kombinierte, gewichtete Ergebnis schlussendlich die besten Ergebnisse. Daher werden das numerische XGBoost-Modell und das textuelle Modell der logistischen Regression im Frontend kombiniert und zu 46.290% textuell und zu 53.710% numerisch gewichtet angebunden. Diese Kombination erreicht einen F1-Score von 0.4220.

4.4 Bewertung der Modelle für die Zielvariable Thema

Die grundsätzlichen Ergebnisse zum Vergleich der Methoden werden auch für die Zielvariable Thema durch den F1-Score bewertet. Dieser ist deutlich höher als bei der Zielvariable Sternzeichen, jedoch trotzdem nicht gut. Bei den textuellen Modellen bewegt er sich etwas über 30%. Dabei erreichte das optimale LSVM-Modell einen Wert von 33,3189%, das optimale Modell der logistischen Regression einen Wert von 32,997% und das optimale XGBoost-Modell einen Wert von 31,592%.

Die Kombination der Hyperparameter, die zu diesen optimalen Modellen führt, ist bei dem LSVM-Modell und dem Modell der logistischen Regression je die Nutzung von TF-IDF und den N-Grams (1,1), beim XGBoost-Modell die Nutzung von TF und den N-Grams (1,1). Daraus folgend lässt sich auch hier kein eindeutiger, genereller Einfluss der verschiedenen Ausprägungen der Hyperparameter erschließen und es wird je Modell die beste Kombination gewählt.

Die numerischen LSVM-Modelle und die Modelle der logistischen Regression erreichen in etwa den gleichen F1-Score wie die textuellen Modelle. Dabei erreicht das LSVM-Modell einen Wert von 32,09% und das Modell der logistischen Regression einen Wert von 32,17%. Jedoch sticht auch hier das XGBoost-Modell positiv mit einem Wert von 47,36% heraus. Dieses ist mit den in Kapitel 3 genannten Hyperparameter mit Abstand das beste Modell. Aus diesem Grund werden das numerische Modell nach XGBoost und das textuelle Modell nach der LSVM mithilfe des ganzen Trainingsdatensatzes trainiert und kommen so auf einen F1-Score von 39,37% für das numerische bzw. 16,32% für das textuelle Modell.

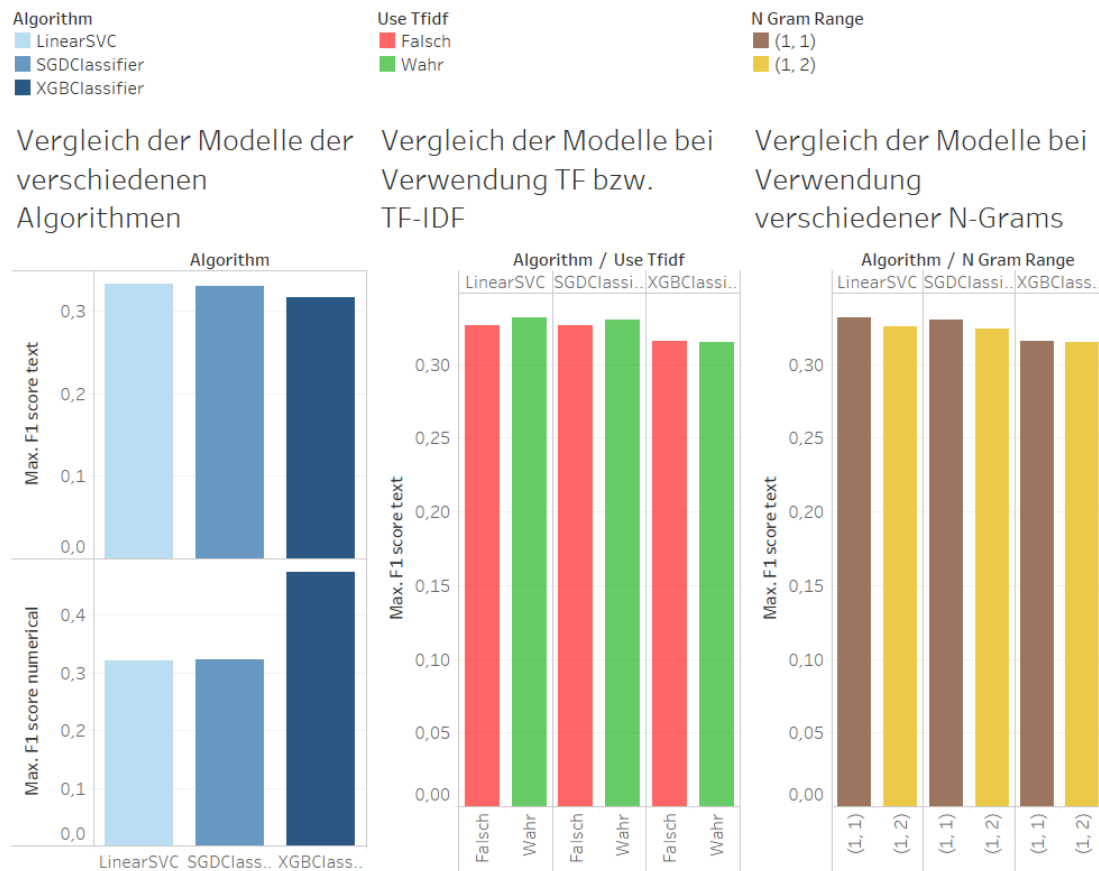


Abbildung 4.4: Bewertung der Modelle für die Zielvariable Thema
Quelle: Eigene Darstellung

Im Frontend wird die Kombination der besten beiden Modelle gewichtet angebunden. Das sind für die Zielvariable Thema das textuelle LSVM-Modell mit 38,95% und das numerische XGBoost-Modell mit 61,05%.

4.5 Bewertung des Clusterings

Auch im Clustering mittels K-means wurden zwei Modelle erstellt, wobei sich diese nicht wie in der Klassifikation gegenseitig gewichten, sondern ergänzen. Beide Ergebnisse werden demnach unabhängig voneinander ausgegeben. Die Güte der jeweiligen Cluster wird durch die Summe der quadrierten Distanzen zwischen den Datenpunkten und dem jeweils nächstgelegenen Clustermittelpunkt berechnet. Für beide Clustermodelle wurde vorab eine Validierung vorgenommen, welche das optimale K bestimmen sollte. Für das textuelle und numerische Clustering wurden dabei jeweils Werte für K zwischen zwei und 40 validiert.

Aus der Validierung ergab sich für das textuelle Modell ein Ellbogenpunkt von 5 und für das numerische Modell ein Ellbogenpunkt von 9 als optimal. Genauere Betrachtungen der Cluster ließen jedoch den Schluss zu, dass ein K von 9 auf dem numerischen Datensatz zu hoch ist. Grund hierfür sind vereinzelte Cluster, welche nur einen Datenpunkt enthalten. Da solche Cluster vermieden werden sollten, um tatsächlich eine Gruppierung aller Datenpunkten zu ermöglichen, musste K soweit reduziert werden, dass nur noch Cluster mit mehr als einem Datenpunkt entstehen. Diese Überlegung überwiegt dabei das Problem, dass ein Modell mit zu wenig Clustern womöglich nicht mehr den nötigen Informationsgehalt liefern kann. Daraus ergab sich letztendlich ein K von 3 für das numerische Clustering. Die entsprechenden Werte für die Summe der quadrierten Distanzen und die Werte für K, welche letztendlich zum Einsatz kamen, sind Grafik 4.5 sind zu entnehmen.

Die in sich homogenen Eigenschaften der Cluster wurden für das textuelle Modell auf Basis einer Textanalyse des ursprünglichen Blogbeitrags und für das numerische Clustering auf Basis einer grafischen Untersuchung von Gemeinsamkeiten und Korrelationen der einzelnen Features vorgenommen. Eine anschließende Interpretation dieser Analysen ergab schließlich die zwei zusätzlichen Features, welche nun ebenfalls für neue Texte vorhergesagt werden können. Die neuen Features können in Tabelle A.2 betrachtet werden.

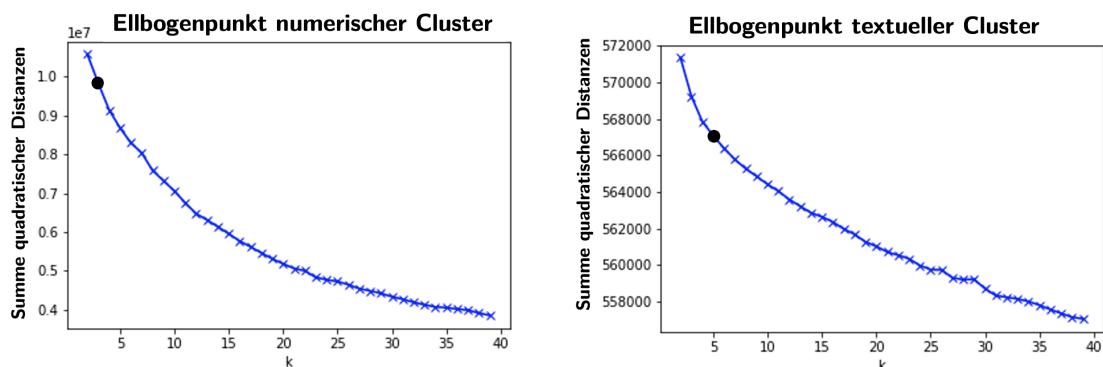


Abbildung 4.5: Validierungsergebnisse des Clusterings
Quelle: Eigene Darstellung

5 Diskussion der Ergebnisse

5.1 Fazit

Abschließend lässt sich aus den in dieser Ausarbeitung generierten Ergebnissen im Vergleich mit der Related Work festhalten, dass weitestgehend ordentliche, jedoch aber keine überragenden Modelle trainiert werden konnten. Dabei sticht vor allem die Zielvariable Sternzeichen negativ und die Zielvariable Thema positiv hervor.

Darüber hinaus ist anzumerken, dass die kombinierte Klassifizierung aller Zielvariablen auf Basis des reinen Textes deutlich schlechter abschneidet als die Klassifizierung nur einer Variablen unter Angabe der Fehlenden. Dies ist nicht überraschend, da dieses Phänomen auch in der Related Work beobachtet werden konnte. Leider entsprechen die numerischen Modelle, deren Ergebnisse durchaus gut abschneiden, aber nicht ganz dem Anwendungskontext, da zu einem zu klassifizierenden normalerweise nicht drei der vier Zielvariablen gegeben sind. Die textuellen Modelle können alle Zielvariablen vorhersagen, performen dabei jedoch nicht zufriedenstellend.

In dieser Arbeit wurden jedoch deutlich mehr Algorithmen und Einstellungen der Hyperparameter zur Erstellung der Modelle betrachtet und evaluiert, daher kann der wissenschaftliche Anspruch durchaus als hoch angesehen werden.

Die Erweiterung durch das Clustering unterstützte leider kaum den zunächst vermuteten Prozess der Erweiterung der Zielvariablen, da nur wenige Cluster identifiziert werden konnten. Diese waren allerdings meist eindeutig von einander abgrenzbar, sodass eine gute Interpretation dieser möglich war. So fokussierte sich beispielsweise eines der entstandenen Cluster auf Verneinungen in den Blogeinträgen, wodurch das zusätzliche Autorenmerkmal des "Negation-Lover" extrahiert werden konnte.

Wie in Kapitel 4 erläutert, sind die textuellen Modelle deutlich schlechter als die numerischen. Der Grund hierfür liegt in der deutlich unterschiedlich großen Dimensionalität der Modelle und dem daraus resultierenden Verhältnis aus Dimensionalität und Datenmenge. Die textuellen Modelle trainierten auf Grundlage von rund 45.000 TF-IDF-Vektoren, die numerischen Modelle hingegen nur auf 71 Dimensionen. Daraus folgt, dass bei einem Trainingsdatensatz von rund 482.000 Einträgen die Generalisierung der numerischen Modelle deutlich besser funktioniert, als die der textuellen, da der Unterschied der Dimensionen deutlich zu erkennen ist. Textmuster sind somit nicht gut approximierbar. Die Muster in den numerischen Modellen sind es jedoch durchaus. Dies könnte allerdings auch durch eine Ausweitung der Datenquelle behoben und deutlich verbessert werden.

5.2 Ausblick

Um das Projekt auch nach Abschluss des Integrationsseminars weiter verbessern zu können, existieren verschiedene Ansätze. Ein deutlich diverserer und größerer Datensatz würde die Modelle voraussichtlich deutlich verbessern. Hierfür müsste der relativ unausgeglichene Datensatz mit zusätzlichen deutlich gleichmäßigeren Daten erweitert werden. Dadurch würde das Thema Student nicht mehr mit Abstand am häufigsten im Datensatz vorkommen. Eine präzisere Vorhersage wäre möglich. Genauso könnte dadurch die aktuell recht begrenzte Altersspanne erweitert werden. Auch die Erhöhung der Datenqualität durch das Ersetzen von Null und unbekannten Werten sollte zu einer Verbesserung der Modelle und dadurch zu einer Verbesserung der Vorhersagen führen.

Darüber hinaus wäre die Anreicherung des Datensatzes um weitere Texttypen eine zusätzliche Verbesserungsmöglichkeit. Auf diese Weise könnten neben Blogeinträgen beispielsweise auch Bücher klassifiziert werden.

Es könnten zudem weitere Clustering Algorithmen erprobt werden, um nicht direkt ersichtliche Gruppen von Autoren zu erkennen und dadurch weitere Merkmale vorhersagen zu können. Dies würde ein komplexeres Bild der Autoren ermöglichen als die derzeitigen Zielvariablen. Ein möglicher Algorithmus hierfür ist die Latent Dirichlet Allocation. Dabei werden Zusammenhänge zwischen den Wörtern betrachtet, um daraufhin die Texte nach diesen Zusammenhängen zu gruppieren.

Abschließend zusammengefasst existieren zahlreiche Optimierungsmöglichkeiten, die jedoch weitestgehend auf der Verbesserung der Datenqualität und -menge basieren und nicht das grundsätzliche Vorgehen in diesem Projekt beeinflussen. Es könnten jedoch auch noch unzählige weitere Algorithmen angewandt und evaluiert werden.

A Tabellen

A.1 Feature Engineering

Feature	Berechnung	Intention
Text Length	Anzahl der Zeichen im Text	Indikator für Textumfang
Number URLs	Anzahl der Worte, die eines oder mehrere der folgenden Wörter enthalten: „urlLink“, „http“, „www“	Indikator dafür, ob auf andere Quellen verlinkt wird.
Number emails	Anzahl der Emails im Text	Sind Ansprechpartner genannt worden?
Uppercase ratio	Anteil der Großbuchstaben an allen Zeichen	Quantifizierung der Zeichenwahl
Lowercase ratio	Anteil der Kleinbuchstaben an allen Zeichen	Quantifizierung der Zeichenwahl
Number ratio	Anteil der Zahlen an allen Zeichen	Quantifizierung der Zeichenwahl
Symbol ratio	Anteil der Sonderzeichen an allen Zeichen	Quantifizierung der Zeichenwahl
Average letter per words	Durchschnitt der Buchstaben pro Wort	Quantifizierung der Wortlänge
Variance of letters per words	Varianz der Wortlänge	Quantifizierung der Heterogenität der Wortlänge
Unique words ratio	Anzahl der verschiedenen verwendeten Wörter im Text	Quantifizierung der Heterogenität der Wortlänge
Average letters per sentence	Durchschnitt der Buchstaben pro Satz	Quantifizierung der Satzlänge
Average letters per sentence	Durchschnitt der Buchstaben pro Satz	Quantifizierung der Satzlänge
Variance of letters per sentence	Varianz der Buchstaben pro Wort	Quantifizierung der Heterogenität der Satzlänge
Average words per sentence	Varianz der Buchstaben pro Wort	Quantifizierung der Worte pro Satz
Variance of words per sentence	Varianz der Worte pro Satz	Quantifizierung der Heterogenität der Worte pro Satz
Maximum uppercase ratio per sentence	Anteil der Großbuchstaben in dem Satz mit dem höchsten Anteil an Großbuchstaben	Zeigt, ob Sätze mit hohem Anteil an Großbuchstaben vorhanden sind
Length of the max. uppercase ratio sentence	Anzahl der Zeichen in dem obigen Feature	Indikator für die Zuverlässigkeit des obigen Features

Tabelle A.1: Durch das Feature Engineering erarbeitete Features

A.2 Clustering-Features

Nummer der Cluster	Interpretation der textbasierten Cluster	Interpretation der numerisch basierten Cluster
0	Up-to-date Person	Explanatory Author
1	Average Citizen.	Hobby Publisher
2	Negation-Lover	Daily Writer
3	Self-referred Author	-
4	Egocentric Person	-

Tabelle A.2: Interpretationen der textuellen und numerisch basierten Cluster

Literaturverzeichnis

- [1] Nick Anstead und Ben O'Loughlin. „Social media analysis and public opinion: The 2010 UK general election“. In: *Journal of Computer-Mediated Communication* 20.2 (2015), S. 204–220.
- [2] Shlomo Argamon et al. „Automatically profiling the author of an anonymous text“. In: *Communications of the ACM* 52.2 (2009), S. 119–123.
- [3] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. New York: Springer, 2006.
- [4] dansbecker. „Using Categorical Data with One Hot Encoding“. In: *Kaggle* (2018). URL: <https://www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding>.
- [5] M. Sudheep Elayidom et al. „Text Classification For Authorship Attribution Analysis“. In: *Advanced Computing: An International Journal* 4 (Okt. 2013). DOI: 10.5121/acij.2013.4501.
- [6] Mehwish Fatima et al. „Multilingual SMS-based author profiling: Data and methods“. In: *Natural Language Engineering* 24.5 (2018), S. 695–724. DOI: 10.1017/S1351324918000244.
- [7] Abdulrahman I Al-Ghadir und Aqil M Azmi. „A study of arabic social media users—posting behavior and author's gender prediction“. In: *Cognitive Computation* 11.1 (2019), S. 71–86.
- [8] Jeff Hale. „Scale, Standardize, or Normalize with Scikit-Learn“. In: *Towardsdatascience* (2019). URL: <https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02>.
- [9] Seifeddine Mechti et al. „A decision system for computational authors profiling: From machine learning to deep learning“. In: *Concurrency and Computation: Practice and Experience* (2020), e5985.
- [10] Pashutan Modaresi, Matthias Liebeck und Stefan Conrad. „Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016.“ In: *CLEF (Working Notes)*. 2016, S. 970–977.

- [11] Todd K Moon, Peg Howland und Jacob H Gunther. „Document author classification using generalized discriminant analysis“. In: *Universidad del Estado de Utah. Estados Unidos* (2006).
- [12] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. Cambridge, MA: MIT Press, 2012.
- [13] Tadashi Nomoto. „Classifying library catalogue by author profiling“. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 2009, S. 644–645.
- [14] Francisco Rangel und Paolo Rosso. „Use of language and author profiling: Identification of gender and age“. In: *Natural Language Processing and Cognitive Science* 177 (2013).
- [15] Francisco Rangel et al. „Overview of the 2nd author profiling task at pan 2014“. In: *CEUR Workshop Proceedings*. Bd. 1180. CEUR Workshop Proceedings. 2014, S. 898–927.
- [16] K. Santosh et al. „Author Profiling: Predicting Age and Gender from Blogs Notebook for PAN at CLEF 2013“. In: *CLEF*. 2013.
- [17] Fabrizio Sebastiani. „Machine learning in automated text categorization“. In: *ACM Computing Surveys* 34.1 (März 2002), S. 1–47. ISSN: 1557-7341. DOI: 10.1145/505282.505283. URL: <http://dx.doi.org/10.1145/505282.505283>.
- [18] Efstathios Stamatatos. „A survey of modern authorship attribution methods“. In: *Journal of the American Society for information Science and Technology* 60.3 (2009), S. 538–556.
- [19] Sharmila Devi V et al. „KCE_DAlab@MAPonSMS-FIRE2018: Effective word and character-based features for Multilingual Author Profiling“. In: *Working Notes of FIRE 2018 - Forum for Information Retrieval Evaluation, Gandhinagar, India, December 6-9, 2018*. Hrsg. von Parth Mehta et al. Bd. 2266. CEUR Workshop Proceedings. CEUR-WS.org, 2018, S. 213–222. URL: <http://ceur-ws.org/Vol-2266/T4-2.pdf>.