Klassifizierung von Spam-E-Mails durch Feature Engineering

Fabian Forthmann
Student Angewandte Informatik
Nordakademie
25337 Elmshorn, Deutschland
fabian.forthmann@nordakademie.de

Ridvan Cetin
Student Angewandte Informatik
Nordakademie
25337 Elmshorn, Deutschland
ridvan.cetin@nordakademie.de

Abstract

Dieses wissenschaftliche Paper beschäftigt sich mit Spam-Klassifizierung mithilfe von Feature Engineering. Es wird davon ausgegangen, dass Features, welche den Inhalt von E-Mails prüfen, besonders geeignet zur Identifizierung von Spam sind. Dabei wurde mithilfe von WEKA Feature Engineering betrieben und als Algorithmus das K-Star Algorithmus verwendet. Features, welche die Struktur von E-Mails überprüfen, haben sich besser bewährt als Features, die den Inhalt prüfen. Während des Trainings konnte das Modell bis zu 93% und auf dem Testdatensatz angewendet 91% der Instanzen korrekt klassifizieren.

1 Einleitung

Durch die starke Verbreitung von E-Mails und den damit verbundenen Anstieg von Spam-Mails [1] ist die Aufgabe Spam-Mails zu erkennen relevanter und komplexer geworden [2]. Für den Nutzer einer E-Mail-Adresse ist es wichtig, nur Nachrichten zu bekommen, die für ihn relevant sind und keine Bedrohung darstellen. Eine erfolgreiche Spam-Klassifizierung verhindert, dass die E-Mail-Adresse einer Person auf Grund von nicht gewollten E-Mails nur eingeschränkt nutzbar ist [1, 3]. Mittlerweile sind Spam-Mails eine der größtmöglichen Sicherheitslücken von Computersystemen [4]. Spam-Mails können eine Bedrohung darstellen, weil sie im Zweifelsfall dem Nutzer Schaden zufügen. Viele dieser E-Mails beinhalten Virenanhänge, Spyware-Agenten, Trojaner oder andere schädliche Software [1, 4, 5]. Wegen dieser Bedrohung ist es wichtig, Spam-Mails möglichst genau zu bestimmen und eine Zustellung zu verhindern.

Dieses Paper entwirft einen Lösungsvorschlag zur Klassifizierung von Spam-Mails mit Hilfe von Machine Learning. Zum Trainieren des Modells wird das Programm WEKA[1] verwendet, welches vor allem auf das Entwerfen von Features angewiesen ist. Es ist davon auszugehen, dass besonders auf den Inhalt der E-Mail bezogene Features einen großen Mehrwert zur Spam-Klassifizierung beitragen.

In dieser Arbeit wird im ersten Schritt E-Mail-Spam mit vorherigen Vorgehensweisen zur Erkennung von Spam in Verbindung gebracht, um daraufhin anhand des vorgestellten Datensatzes die Entwicklung der Features und das Training des Modells zu erläutern. Abschließend werden die Ergebnisse diskutiert und Erkenntnisse aus dem Training zusammengefasst.

2 E-Mail-Spam

Unter Spam sind Nachrichten, Beiträge oder Inhalte zu verstehen, die von dem Empfänger oder Betreiber eines Angebots nicht gewollt sind und unaufgefordert erstellt oder versendet werden. Spam ist neben E-Mails auch bei Online-Rezensionen, in Social Media, bei Blogeinträgen oder bei der Erstellung von Spam-Blogs zu beobachten. [6] E-Mail-Spam bezeichnet ungewollte und unaufgefordert versendete E-Mails zu dessen Verfasser der Empfänger keine Beziehung hat. Solche E-Mails werden vor allem aus monetären Gründen versendet. Die Verfasser profitieren entweder durch die verschickte Werbung oder durch kriminelle Angriffe durch die versendeten E-Mails. Erwünschte E-Mails werden als Ham bezeichnet [4]. [5, 6]

Die Erfassung von Spam-Mails durch Spam-Filter ist technisch durch unterschiedliche Methoden möglich. Neben dem hier vorgestellten

[1] https://www.cs.waikato.ac.nz/ml/weka/

Vorgehen mit Machine Learning und Feature Engineering wurden in der Vergangenheit vor allem regelbasierte Filter und Filter basierend auf Deep Learning entwickelt. [7]

3 Verwandte Arbeiten

Die Klassifizierung von Spam besitzt eine große Relevanz und ist somit auch breit erforscht. Die Forschung bezieht sich nicht ausschließlich auf E-Mail-Spam. In [8] untersuchen Michael Crawford et al. wie Spam-Rezensionen bei Online-Marktplätzen durch Supervised Learning identifiziert werden können. In ihrem Paper kommen die Autoren zu ähnlichen Ergebnissen wie Draško Radovanović et al. in [9]. Pranam Kolari et al. untersuchen in ihrem Paper Spam-Blogs die durch ihren Inhalt Such-Algorithmen in die Irre führen und identifizieren sie durch die Entwicklung von Features [10]. In [11] klassifizieren Dengyyong Thou et al. Web-Spam durch gerichtete Graphen. Andere Arbeiten wie die von Darshit Pandya beschäftigen sich mit Spam-Nachrichten von Messaging- und E-Mail-Diensten [12]. Dabei zeigt Pandya Parallelen zwischen SMS- und E-Mail-Spam auf und klassifiziert diese durch eine Support Vector Machine.

Die Erkennung von E-Mail-Spam im Kontext von Machine Learning wurde in vergangenen Arbeiten vor allem durch das Trainieren eines Modells mit gelabelten Daten vorangebracht. Zur Klassifizierung der Mails werden unterschiedliche Methoden wie ein Random Forest [1, 4], Support Vector Machines [12], Vektorraum-Retrieval [13], Entscheidungsbäume [2] oder Bayessche Netze [3] verwendet. Dabei zeichnet sich ab, dass besonders Support Vector Machines zu sehr guten Ergebnissen der Spam-Klassifizierung gelangen [10, 12].

4 Daten

Zum Trainieren des hier vorgestellten Modells wurden die Trainingsdaten aus [14] verwendet. Sie sind online unter folgender Adresse zu finden:

https://github.com/benjamincohen1/SpamClassifierOfficeHours

Der Datensatz besteht aus 902 englischsprachigen E-Mails, die von dem Herausgeber als Spam oder Ham gelabelt wurden. Insgesamt beinhalten die Trainingsdaten 443 Spam- und 459 Ham-Mails. Der verwendete Datensatz wurde nicht bereinigt und beinhaltet somit nicht nur reinen Fließtext, sondern auch HTML-Inhalte. Ein Datum beinhaltet neben einem Label den E-Mail-Text und den Betreff der Mail. Insgesamt bestehen die verwendeten Trainingsdaten aus 265.818 Wörter in 27.358 Sätzen mit 2.341.189 Buchstaben.

5 Methodik

Ziel dieser Arbeit ist es, durch das Training eines Machine-Learning-Modells E-Mail-Spam und E-Mail-Ham in 92% der Fälle richtig klassifizieren zu können. Dafür wird nach dem Training mit dem vorgestellten Trainingsdatensatz das Modell an einem Testdatensatz mit 100 Instanzen validiert.

5.1 Pre-Processing

Weil der Datensatz keine ganzheitlich unbrauchbaren E-Mails beinhaltet und vollständig gelabelt wurde, musste keine Bereinigung von ganzen E-Mails vorgenommen werden. Es wurde auch auf Normalisierungen in den einzelnen Mails verzichtet, weil in der Feature-Entwicklung nicht nur der Textinhalt, sondern auch Metadaten berücksichtigt wurden. Somit wurden beispielsweise keine Sonderzeichen und HTML-Stücke entfernt oder Groß- und Kleinschreibung verändert. Bei E-Mails mit Encoding-Fehlern wurden diese behoben bevor der Datensatz zum Training verwendet wurde.

5.2 Feature Engineering

Die verwendeten Features wurden vor allem auf Grundlage von bekannten Eigenschaften von Spam-Mails entwickelt. Nach der Entwicklung wurden für das endgültige Modell folgende Features ausgewählt, die in ihrer Kombination die höchste Anzahl von korrekt identifizierten Instanzen geliefert haben.

Wörter in Versalien Index (`cap_word_ratio`). Ein Großteil von Spam-E-Mails arbeitet mit verhältnismäßig vielen Wörtern, die ganz großgeschrieben sind, um die Aufmerksamkeit des Lesers zu steigern oder das Gefühl von hoher Dringlichkeit zu erzeugen. Dieses Feature stellt die Anzahl von vollkommen groß geschriebenen Wörtern im Verhältnis zur Textlänge dar.

Wörter in Versalien Anzahl (`amount_cap_words`). Als Ergänzung zu dem oben genannten Index zählt dieses Feature die Anzahl von

vollkommen groß geschriebenen Wörtern in dem Text.

Anzahl Großbuchstaben (`amount_capitalized_letters`). Neben der Anzahl von vollkommen groß geschriebenen Wörtern kann es relevant sein, wie viele einzelne Buchstaben in einer E-Mail großgeschrieben werden. Durch dieses Feature wird gezählt, wie viele Buchstaben eines Textes großgeschrieben sind.

HTML-Tags Anzahl (`html_tag_count`). Viele Spam-Mails bestehen nicht nur aus Text, sondern auch aus visuellen Elementen, die durch HTML-Code zu erkennen sind. Hier wird gezählt wie oft in einer E-Mail die Zeichen „<" oder „>" vorkommen.

HTML-Vorkommen (`has_html`). Ein weiteres Anzeichen für HTLM-Teile in einer E-Mail ist, dass „`html`" als Text vorkommt. Dieses Feature setzt eine Flag, falls das Wort „`html`" in dem ausgewählten Text vorkommt. Dabei wird Groß- und Kleinschreibung ignoriert.

Anzahl eckige Klammern (`square_bracket_count`). Ein Zeichen, welches in HTML oft benutzt wird, sind eckige Klammern. Dieses Feature zählt wie oft „`[`" oder „`]`" in einer E-Mail vorkommen.

Anzahl Links (`num_link`). In vielen Fällen wollen Spam-Mails den Nutzer auf eine Website weiterleiten. Diese wird über Links in die E-Mail eingebettet. Dieses Feature identifiziert die Anzahl von Verlinkungen, indem es zählt, wie oft „`http`" in einer Instanz vorkommt.

Titel mit „re" (`begin_with_re`). Die wenigsten Spam-Mails probieren als Antwort auf eine Mail zu erscheinen. Dafür sind ein Großteil von Ham-Mails Antworten auf vorherige E-Mails. Hier wird ein Flag gesetzt, falls das erste Wort im Titel einer E-Mail „`re`" ist. Dabei wird Groß- und Kleinschreibung ignoriert.

Anzahl Ausrufezeichen (`count_exclamation_marks`). Ähnlich wie bei der Anzahl von Großbuchstaben probieren Verfasser von Spam-Mails durch die starke Verwendung von Ausrufezeichen ein Gefühl von Dringlichkeit und Wichtigkeit zu vermitteln. Bei diesem Feature wird die Anzahl von Ausrufezeichen in einer Instanz gezählt.

Durch exploratives Untersuchen aller Features wurden andere Features nicht in das Trainings-Modell aufgenommen, weil sie zu keiner Verbesserung der richtig klassifizierten Instanzen geführt haben. Die nicht berücksichtigten Features haben den Text nach Spam- oder Ham-typischen Wörtern untersucht, die Anzahl von Wörtern einer Mail ermittelt und die E-Mail nach Fotos oder GIFs überprüft.

5.3 Modell

Obwohl Deep Learning ein aktueller Ansatz zur Klassifizierung von Spam ist, der eine hohe Präzision bei der Klassifizierung von Spam erreicht, sind die Entscheidungen der Modelle weitestgehend nicht nachvollziehbar [15, 16].

Daher wurde in diesem Paper ein Machine Learning Ansatz verwendet. Es wurden unterschiedliche Algorithmen, wie beispielsweise die Lineare Regression, der Naive Bayes, der LMT und der K-Star Algorithmus, zum Training des Modells verwendet. Dabei erzielte der K-Star Algorithmus die größte Anzahl von richtig Klassifizierten Instanzen.

Bei dem K-Star Algorithmus handelt es sich um einen instanzbasierten Klassifikator. Bei diesem Algorithmus basiert die Klasse einer Testinstanz auf einer Klasse der ihr ähnlichen Trainingsinstanz. Mithilfe einer Ähnlichkeitsfunktion ist es dann möglich, zu bestimmen, welche Instanzen sich ähnlich sind [17]. Dieser Algorithmus ist den Lazy-Klassifikatoren zuzuordnen und ist für die Clusteranalyse entwickelt worden. [18]

Zur Trainierung des Modells wurde eine 10-Fold Cross-Validation verwendet, um möglichst viele Kombinationen aus den Trainingsdaten generieren zu können.

6 Ergebnisse

Das entwickelte Modell, welches neun Attribute enthält, ist am Ende des Trainings in der Lage, 93,2737% der Trainingsinstanzen korrekt zu klassifizieren. 6,7627% der Instanzen werden falsch klassifiziert.

Tatsächlicher Wert	Klassifiziert als: Ham	Klassifiziert als: Spam
Ham	431	28
Spam	33	410

Abbildung 1: Confusion Matrix des Trainingsdatensatzes

Anhand der Confusion Matrix des Modells während des Trainings (siehe Abbildung 1) ist zu erkennen, dass von den 459 Ham-Daten 431 korrekt als True-Negative und 28 als False-Positive identifiziert werden. Von den 443 Spam-Daten werden am Ende des Trainings 33 als False-Negative und 410 als True-Positiver klassifiziert.

Das Modell konnte nach Anwendung des Testdatensatzes insgesamt 91% der Instanzen korrekt klassifizieren.

		Klassifiziert als: Ham	Klassifiziert als: Spam
Tatsächlicher Wert	Ham	37	5
	Spam	4	54

Abbildung 2: Confusion Matrix des Testdatensatzes

Wie an der Confusion Matrix des Testdatensatzes (siehe Abbildung 2) zu erkennen ist, wurden von den 42 Ham-Daten 37 als True-Negative und 5 als False-Positive bestimmt. Von den 58 Spam-Daten wurden 4 als False-Negative und 54 als True-Positive erkannt.

Mithilfe der Daten aus der Confusion Matrix ist es möglich, weitere Angaben zu berechnen (siehe Abbildung 3). Alle weiteren Angaben entsprechen dem gewichteten Durchschnitt. Die TP-Rate (True-Positive) beträgt 0,91 und die FP-Rate (False-Positive) beträgt 0,098. Die Precision und der Recall betragen beide 0,91. Somit liegt der F1-Wert ebenfalls bei 0,91. Der ROC Area Wert liegt bei 0,9803.

7 Diskussion

Während des Trainings kann das Modell die Instanzen des Trainingsdatensatzes zu 93% korrekt klassifizieren. Damit übertrifft das Modell im Training das Ziel dieses Papers um 1%. Nach der Anwendung des Modells auf den Testdatensatz ist anhand der Ergebnisse zu erkennen, dass sich das Modell um knapp zwei Prozent verschlechtert hat. Eine kleine Verschlechterung ist nicht ungewöhnlich [19]. Dennoch kann es dafür verschiedene Gründe wie einen nicht balancierten Testdatensatz oder ein minimales Overfitting geben.

Auffällig ist, dass die Daten sowohl beim Trainingsdatensatz als auch beim Testdatensatz eine Dysbalance aufweisen. Bei dem Trainingsdatensatz gibt es mehr Ham als Spam und bei dem Testdatensatz gibt es mehr Spam als Ham. Wobei zu erwähnen ist, dass die Dysbalance des Trainingsdatensatzes um einiges stärker ausgeprägt ist als beim Testdatensatz. Während beim Trainingsdatensatz ungefähr 49,2% Spam sind, sind es im Testdatensatz 58%. Diese Dysbalance kann dazu geführt haben, dass das Modell in Verbindung mit dem Testdatensatz 2% weniger Instanzen richtig klassifiziert hat. [19]

Eine weitere Ursache für das etwas schlechtere Test-Ergebnis kann ein geringes Overfitting sein. Ein Overfitting des Modells kann dadurch zustande kommen, dass Features vor allem auf dem Trainingsdatensatz getestet und darauf angepasst wurden. [20, 21]

Bei dem endgültigen Modell lag der Recall-Wert im Durchschnitt bei 0,932 (siehe Abbildung 4). Zum Ende des Trainings machte das Modell sogar weniger Fehler bei der Klassifizierung von Ham als bei Spam. Es wurden also weniger Instanzen von Ham fälschlicherweise klassifiziert als von Spam. Somit konnte das Modell auf den Trainingsdatensatz angewendet besser Spam als Ham erkennen und machte weniger Fehler. Bei dem Testdatensatz liegt der Recall-Wert im Durchschnitt bei 0,910 und ist somit etwas schlechter als während der Trainingsphase. Außerdem ist das Modell beim Testdatensatz besser darin, Spam als Ham zu identifizieren und identifizierte mehr Spam-Instanzen richtig. Daher ist davon auszugehen, dass eine zweite Ursache für die Verschlechterung ein minimales Overfitting des Modells ist. Die Werte Precision und F-Wert

Klasse	TP-Rate	FP-Rate	Precision	Recall	F-Measure	ROC Area
Ham	0,881	0,069	0,902	0,881	0,892	0,980
Spam	0,931	0,119	0,915	0,931	0,923	0,980
Durchschnitt	0,910	0,098	0,910	0,910	0,910	0,980

Abbildung 3: Detaillierte Darstellung des Modells auf den Testdatensatz

Klasse	TP-Rate	FP-Rate	Precision	Recall	F-Measure	ROC Area
Ham	0,939	0,074	0,929	0,939	0,934	0,980
Spam	0,926	0,061	0,936	0,926	0,931	0,980
Durchschnitt	0,932	0,068	0,932	0,932	0,932	0,980

Abbildung 4: Detaillierte Darstellung des Modells auf den Trainingsdatensatz

liegen außerdem unter dem Ergebnis der Trainingsphase und deuten auf ein minimales Overfitting hin.

Dennoch muss gesagt werden, dass es bei dieser minimalen Verschlechterung schwierig ist, eine Aussage darüber zu treffen, wodurch der Performance-Verlust begründet ist. Um die genauen Ursachen der Verschlechterung zu erfahren, erfordert es eine aufwändige Recherche und Analyse des Modells und der Daten. Jedoch würde eine solche Analyse den Rahmen dieses Papers überschreiten. Die genannten Gründe sind nur mögliche Gründe für die Verschlechterung.

Die für das endgültige Modell verwendeten Features prüfen nicht auf den Inhalt eines Textes, sondern vor allem die Struktur und die Metadaten einer Mail. So wird in dem jetzigen Modell viel mehr auf Großschreibung, bestimmte Elemente und den Aufbau einer E-Mail geachtet. Dieser, unserer Hypothese widersprechenden Feature-Auswahl, ist die starke Genauigkeit des Modells zu verdanken.

Mit 91% richtig klassifizierten Instanzen hat sich das hier vorgestellte Modell bewährt, auch wenn es die Zielmarke von 92% mit dem Testdatensatz nicht erreich hat. Die starke Genauigkeit des Modells wird außerdem durch den ROC-Wert unterstützt, der bei 0,9803 liegt (siehe Abbildung 5). Dieser hohe Wert deutet darauf hin, dass es sich hierbei um ein gutes Modell handelt. Wie in Abbildung 5 zu erkennen ist, verläuft die ROC-Kurve über die linke obere Ecke, was auf ein treffsicheres Modell hindeutet.

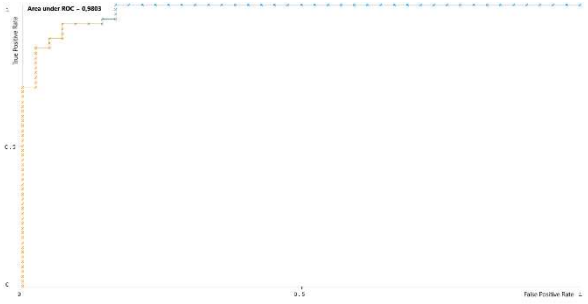

Abbildung 5: ROC-Kurve

Außerdem ist zu beachten, dass sich die anderen Werte, wie Recall, F1, Precision, TP-Rate und FP-Rate, nur minimal von den Ergebnissen des Trainingsdatensatzes unterscheiden. Es war jedoch ein minimaler Unterschied zwischen Trainings- und Testergebnis zu erwarten, weil das üblich ist [21].

8 Fazit

In dieser Arbeit wurde gezeigt, dass mithilfe von Feature Engineering und Machine Learning eine gute Klassifizierung von Spam und Ham möglich ist, weil mit dem entwickelten Modell 91% der Instanzen im Testdatensatz richtig klassifiziert wurden. Dabei ist auffällig, dass vor allem Features, die sich auf die Metadaten der E-Mails beziehen, einen großen Wissensgewinn für das Modell generieren. In künftigen Arbeiten wäre es interessant zu untersuchen, ob weitere Metadaten wie das Sendedatum oder das Herkunftsland eine Aussage über den Inhalt der Mail treffen können.

Quellen

[1] S. M. Lee, D. S. Kim, J. H. Kim und J. S. Park, „Spam Detection Using Feature Selection and Parameters Optimization“ in 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), Krakow, TBD, Poland, 15.02.2010-18.02.2010, S. 883–888, doi: 10.1109/CISIS.2010.116.

[2] O. El Kouari, H. Benaboud und S. Lazaar, „Using machine learning to deal with Phishing and Spam Detection“ in NISS2020: The 3rd International Conference on Networking, Information Systems & Security, Marrakech Morocco, 03312020, S. 1–7, doi: 10.1145/3386723.3387891.

[3] S. K. Trivedi, „A study of machine learning classifiers for spam detection“ in 2016 4th International Symposium on Computational and Business Intelligence (ISCBI), Olten,

Switzerland, 05.09.2016 - 07.09.2016, S. 176–180, doi: 10.1109/ISCBI.2016.7743279.

[4] Dave DeBarr und Harry Wechsler, „Spam Detection using Clustering, Random Forests, and Active Learning“, CEAS 2009 – Sixth Conference on Email and Anti-Spam, 2009.

[5] M. Iqbal, M. M. Abid, M. Ahmad und F. Khurshid, „Study on the Effectiveness of Spam Detection Technologies“, IJITCS, Jg. 8, Nr. 1, S. 11–21, 2016, doi: 10.5815/ijitcs.2016.01.02.

[6] G. V. Cormack, Email spam filtering: A systematic review. Boston: Now Publishers, 2008.

[7] Ankit Narendrakumar Soni, „Spam e-mail detection using advanced deep convolution neural network algorithms“, Journal for innovative development in pharmaceutical and technical science, Nr. 2, 74-80, 2019, Art. no. 5.

[8] M. Crawford, T. M. Khoshgoftaar, J. D. Prusa, A. N. Richter und H. Al Najada, „Survey of review spam detection using machine learning techniques“, Journal of Big Data, Jg. 2, Nr. 1, 2015, doi: 10.1186/s40537- 015-0029-9.

[9] D. Radovanovic und B. Krstajic, „Review spam detection using machine learning“ in 2018 23rd International Scientific-Professional Conference on Information Technology (IT), Zabljak, 19.02.2018 - 24.02.2018, S. 1–4, doi: 10.1109/SPIT.2018.8350457.

[10] Pranam Kolari, Akshay Java, Tim Finin, Tim Oates und Anupam Joshi, „Detecting Spam Blogs: A Machine Learning Approach“, NSF Awards American Association for Artificial Intelligence, 1352 - 1356, 2006.

[11] D. Zhou, C. J. C. Burges und T. Tao, „Transductive link spam detection“ in the 3rd international workshop, Banff, Alberta, Canada, 2007, S. 21, doi: 10.1145/1244408.1244413.

[12] D. Pandya, „Spam Detection Using Clustering-Based SVM“ in the 2019 2nd International Conference, Jakarta, Indonesia, 2019, S. 12–15, doi: 10.1145/3366750.3366754.

[13] M. Sasaki und H. Shinnou, „Spam detection using text clustering“ in 2005 International Conference on Cyberworlds (CW'05), Singapore, 23.11.2005 - 25.11.2005, 4 pp-319, doi: 10.1109/CW.2005.83.

[14] Benjamin Cohen, SpamClassifierOfficeHouse. [Online]. Verfügbar unter: https://github.com/benjamincohen1/SpamClassifierOfficeHours (Zugriff am: 9. Dezember 2020).

[15] L. Deng, „Deep Learning: Methods and Applications“, FNT in Signal Processing, Jg. 7, 3-4, S. 197–387, 2014, doi: 10.1561/2000000039.

[16] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi und O. E. Ajibuwa, „Machine learning for email spam filtering: review, approaches and open research problems“ (eng), Heliyon, Jg. 5, Nr. 6, e01802, 2019, doi: 10.1016/j.heliyon.2019.e01802.

[17] John G. Cleary, Leonard E. Trigg, „K*: An Instance-based Learner Using an Entropic Distance Measure“, 12th International Conference on Machine Learning, S. 108–114, 1995.

[18] Mohan Vijayarani, „Comparative Analysis of Bayes and Lazy Classification Algorithms“.

[19] N. V. Chawla, N. Japkowicz und A. Kotcz, „Editorial“, SIGKDD Explor. Newsl., Jg. 6, Nr. 1, S. 1–6, 2004, doi: 10.1145/1007730.1007733.

[20] A. Singh, N. Thakur, A. Sharma, „A review of supervised machine learning algorithms“, 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), S. 1310–1315, 2016.

[21] H. Wu und J. L. Shapiro, „Does overfitting affect performance in estimation of distribution algorithms“ in the 8th annual conference, Seattle, Washington, USA, 2006, S. 433, doi: 10.1145/1143997.1144078.

A Arbeitsteilung

Die verwendeten Features wurden von Ridvan Cetin und Fabian Forthmann gemeinsam entwickelt. Nachfolgend ist aufgeführt wer welches Kapitel des Papers hauptverantwortlich verfasst hat.

Abstract: Ridvan Cetin
1 - Einleitung: Fabian Forthmann
2 - E-Mail-Spam: Fabian Forthmann
3 - Verwandte Arbeiten: Fabian Forthmann
4 - Daten: Fabian Forthmann
5 - Methodik: Fabian Forthmann
5.1 - Pre-Processing: Fabian Forthmann
5.2 - Feature Engineering: Fabian Forthmann
5.4 - Modell: Ridvan Cetin
6 - Ergebnisse: Ridvan Cetin
7 - Diskussion: Ridvan Cetin
8 - Fazit: Fabian Forthmann

Die in diesem Paper vorgestellten Features wurden auf einen GitHub-Repository entwickelt. Es ist unter folgender Adresse zu finden:

https://github.com/FForthmann/spam-detection-WPK-2020