Klassifizierung von E-Mails mittels Machine Learning

SECANA - Gruppe 5

30. Juni, 2021

Version: Abgabe

Hochschule Albstadt-Sigmaringen
Fakultät Informatik

Projektarbeit

Klassifizierung von E-Mails mittels Machine Learning

SECANA - Gruppe 5

Prüfer Prof. Dr. German Nemirovski
Fakultät Informatik
Hochschule Albstadt-Sigmaringen

30. Juni, 2021

SECANA - Gruppe 5
Klassifizierung von E-Mails mittels Machine Learning
Projektarbeit, 30. Juni, 2021
Prüfer: Prof. Dr. German Nemirovski

Hochschule Albstadt-Sigmaringen
Fakultät Informatik
Poststraße 6
72458 Albstadt

Zusammenfassung

Das Ziel dieses Projektes war das Erstellen einer Anwendung zur Klassifizierung von Spam-Emails mittels Machine-Learning-Technologie. Hierzu wurde nach einer ersten Reinigung und Vorverarbeitung von bereits klassifizierten E-Mail-Datensätzen unter Verwendung verschiedener Algorithmen diverse Modelle in Kombination mit den gewünschten Features entwickelt, welche im Anschluss auf ihre Genauigkeit verglichen wurden. Für eine praktische Anwendung des Modells wurde eine Web-Anwendung in Flask entwickelt, welche eine Spam-Überprüfung im Browser ermöglicht. Ergänzend wurde ein Plugin für das Mailprogramm „Mozilla Thunderbird“, zur direkten Überprüfung in einem nativen User-Interface, angefertigt. Diese wurden anschließend zur Analyse von neu gesammelten Spam-E-Mails verwendet, um die Richtigkeit der Klassifizierungsfähigkeit des Modells zu überprüfen. Mittels Testdaten konnten, je nach Datensatz, Präzisionswerte von 56% bis 76% erzielt werden. Um die Spanne bei den Genauigkeiten zu verstehen, wurden verschiedene Faktoren und deren Auswirkungen auf das Resultat untersucht.

Abstract

The goal of this project was to create an application for the classification of spam emails using machine learning technology. For this purpose, after an initial cleaning and pre-processing of already classified email datasets using different algorithms, various models were developed in combination with the desired features. These models were then compared with respect to their accuracy. For a practical application of the model, a web application was developed in Flask, which allows for a spam check in the browser. Additionally, a plugin for the mail program "Mozilla Thunderbird", for direct checking in a native user interface. These were then used to analyze newly collected spam e-mails, to check the correctness of the classification ability of the model. Using test data, precision values of 56% to 76% were achieved, depending on the data set. To understand the range in accuracies, various factors and their effects on the results were examined.

Eigenständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit eigenständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie die aus fremden Quellen direkt oder indirekt übernommenen Stellen / Gedanken als solche kenntlich gemacht haben. Diese Arbeit wurde noch keiner anderen Prüfungskommission in dieser oder einer ähnlichen Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

Hiermit stimmen wir zu, dass die vorliegende Arbeit von der Prüferin / dem Prüfer in elektronischer Form mit entsprechender Software auf Plagiate überprüft wird.

Albstadt, 30. Juni, 2021

SECANA - Gruppe 5

Inhaltsverzeichnis

1 Einführung

1.1 Definition

Mit dem Projekt soll die Möglichkeit geschaffen werden, E-Mail Spam automatisiert zu erkennen und als solchen zu klassifizieren. Hierfür ist es wichtig zu beschreiben, wie die beiden Klassen „Spam“ und „Ham“ definiert werden können.

Ham

Bei E-Mails der der Klasse „Ham“ handelt es sich um legitime E-Mails. Diese sind von dem jeweiligen Nutzer gewollt und besitzen keinen maliziösen oder verdächtigen Hintergrund.

Spam

Im Gegensatz zu legitimen E-Mails sind Spam Nachrichten von dem Empfänger nicht gewollt und können diesem potenziell sogar schaden.

Das Bundesamt für Sicherheit in der Informationstechnik beschreibt den Begriff Spam wie folgt:

„Unter Spam versteht man unerwünschte Nachrichten, die massenhaft und ungezielt per E-Mail oder über andere Kommunikationsdienste versendet werden. In der harmlosen Variante enthalten Spam-Nachrichten meist unerwünschte Werbung. Häufig enthält Spam jedoch auch Schadprogramme im Anhang, Links zu verseuchten Webseiten oder wird für Phishing-Angriffe genutzt.“ [1]

1.2 Projektziele

Das Ziel des Projekts ist eine Kategorisierung der E-Mails in den Klassen Spam und Ham, unter Verwendung eines Machine Learning Ansatzes. Für die Genauigkeit der Klassifikation wird ein Wert von mindestens 75% angestrebt.

Es werden dabei nur textuelle Inhalte einer E-Mail, und nicht E-Mail-Anhänge, analysiert. Dadurch kann keine Entscheidung darüber getroffen werden, ob es sich bei einem Anhang um ein valides Dokument oder beispielsweise einen Makrovirus handelt. Das erzeugte Modell zur Spamerkennung soll zudem eine vereinfachte Interaktionsmöglichkeit für Nutzende bieten.

Hauptteil

2

2.1 Entwicklung des Models

In den folgenden Unterkapiteln wird die Vorgehensweise erläutert, mit welcher das Modell zur Spam-Erkennung aufgebaut wurde.

2.1.1 Datasets

Die Grundlage für das Projekt stellen die jeweiligen Datensätze dar, die für die Analyse und für das Training des Modells benötigt werden. Bei der Wahl der Datensätze wurde der Fokus auf englische Vertreter gelegt, da hier eine höhere Anzahl an potenziellen Kandidaten im Vergleich zum Deutschen existierte.

Nach reichlicher Recherche ist die Wahl auf den Datensatz des Open-Source Projektes „Apache SpamAssassin“ gefallen. Dieser kann auf der Projektseite gefunden und heruntergeladen werden. Zu beachten ist, dass dieser in Kategorien wie „easy“ und „hard“ unterteilt wird; im Rahmen des Projektes wurden hier alle genutzt. Enthalten sind etwa 6000 E-Mails - davon sind ca. 31% Spam. Alle enthaltenen E-Mails stammen aus den Jahren 2002 bis einschließlich 2005 [2, 3].

Nach Rücksprache mit dem betreuenden Professor wurde bestätigt, dass für die Qualität eines Modells eine größere Anzahl an Daten notwendig ist. Aus diesem Grund wurden ergänzende Spam- und Ham-Datensätze gesucht und integriert. Ergänzt wurde der initiale Datensatz durch den Datensatz „TREC 2007 Public Corpus“. Dieser enthält etwa 75.000 E-Mails, davon sind ca. 50.000 Spam und ca. 25.000 Ham [4, 5].

Insgesamt waren anschließend 52.596 Spam- und 32.171 Ham-E-Mails für das Projekt verfügbar, was einer 16x größeren Datenmenge im Vergleich zum Beginn darstellt.

2.1.2 Data Preprocessing

Im ersten Schritt des Projektes wurden die im Unterkapitel „Datasets“ gesammelten Daten eingelesen. Genutzt wurde hierfür die Sprache Python in Kombination mit einem Jupyter Notebook mit der JupyterLab-Umgebung. Dieses ermöglicht es, auf übersichtliche Art und Weise, Quellcode und Programmausgaben zu organisieren.

Zu Beginn werden alle Dateipfade der verschiedenen Spam- und Ham-Mails in zwei Listen aufgenommen. Über diese Listen kann später die Parser-Funktion iterieren. Die beiden Listen haben die Namen „all_spam“ (enthält alle Spam Mails) und „all_ham“ (enthält nur legitime Mails).

Zum Einlesen und Verarbeiten der beiden Listen werden die beiden Funktionen *parse_email* und *parse_dataset* genutzt. Die Aufgabe und Funktion dieser wird in den nachfolgenden beiden Blöcken näher beschrieben.

Im einfachsten Beispiel besteht eine E-Mail aus Headern (zusätzlichen Informationen wie Metadaten) und einem Body. Letzterer enthält beispielsweise die eigentlichen, für den Nutzenden sichtbaren Informationen, wie den folgenden Text: „Sehr geehrte Damen und Herren, [...] Viele Grüße Paul“. Eine E-Mail kann zudem aus mehreren Teilen bestehen, die auch unterschiedlich kodiert sein können. Der Inhalt einer Nachricht kann sich dadurch über mehrere Teile erstrecken. Diese Art von E-Mails wird als Multipart bezeichnet.

Parse Emails

parse_email wird dazu verwendet, den Inhalt einer E-Mail zu dekodieren. Die Funktion wird nachfolgend kurz beschrieben.

Die Funktion *parse_email* bekommt eine E-Mail übergeben. Handelt es sich um eine Multipart Mail wird zusätzlich über jede der Teile iteriert. Bei einer Multipart Mail handelt es sich um eine E-Mail, die aus mehreren Teilen besteht [6]. Diese Teile können in verschiedenen kodiert sein. Handelt es sich bei dem Content um „text/html“ (HTML-Basiert) oder „text/plain“ (Plaintext) wird überprüft ob der Inhalt der E-Mail für den Transport als „base64“ oder „quoted-printable“ kodiert wurde. Ist das der Fall, wird dieser wieder dekodiert. Der Inhalt E-Mail wird anschließend zurückgegeben.

Parse Dataset

parse_dataset erfüllt die Aufgabe, alle Spam- und Ham-E-Mails einzulesen und einem Label zuzuweisen. Extrahiert werden Informationen wie Inhalt, Betreff, Absender, URLs und Domains. Die Funktion wird einmal für Spam und Ham ausgeführt. Die Information, wie die E-Mails gelabelt werden sollen, wird dabei übergeben.

Hierfür iteriert die Funktion über alle übergebenen E-Mails und extrahiert durch die Verwendung eines E-Mail Parsers „Betreff“ und „Absender“. Die Erkennung von URLs erfolgt durch einen regulären Ausdruck. Aus diesem wird wiederum mit Hilfe eines URL Parsers versucht, eine Domain auszulesen. Das Einlesen des E-Mail Inhaltes erfolgt durch die Funktion *parse_email*. Handelt es sich dabei um eine HTML basierte E-Mail, so werden die Daten durch einen HTML-Parser extrahiert.

Create Dataframe

Die aus den Parsing-Funktionen erhaltenen Informationen, die bisher noch in Spam und Ham getrennt waren, werden nun in einer Liste zusammengefasst. Diese neue Liste wird nun genutzt, um ein Pandas DataFrame anzulegen. Das DataFrame besteht aus den Spalten „spam“, „raw_data“, „subject“, „from“, „urls“ und „domains“. Zusätzlich gibt es auf der linken Seite eine Zeilennummer, die jeden Eintrag eindeutig repräsentiert.

Aktuell sind die beiden Kategorien „SPAM“ und „HAM“ zwar in einer gemeinsamen Liste, aber noch immer getrennt, da diese nur aneinander angehängt wurden. Um diese Trennung aufzulösen, wird der Inhalt des DataFrame mit einem sogenannten „shuffle“ Befehl durchgemischt. Anschließend müssen noch die Zeilennummern (IDs) zurückgesetzt werden, da auch diese nun nicht mehr aufsteigend angeordnet sind. Die Verteilung ist in Abbildung 2.1 ersichtlich, dabei entspricht „SPAM“ - 1 und „HAM“ - 0.

Im letzten Schritt wird dem DataFrame noch eine neue Spalte hinzugefügt, diese trägt den Namen „chars“ und gibt die Anzahl der Zeichen des Mail Inhaltes für jeden Eintrag an. Das erzeugte DataFrame hat nun die Form, wie sie in Abbildung 2.2 zu sehen ist.

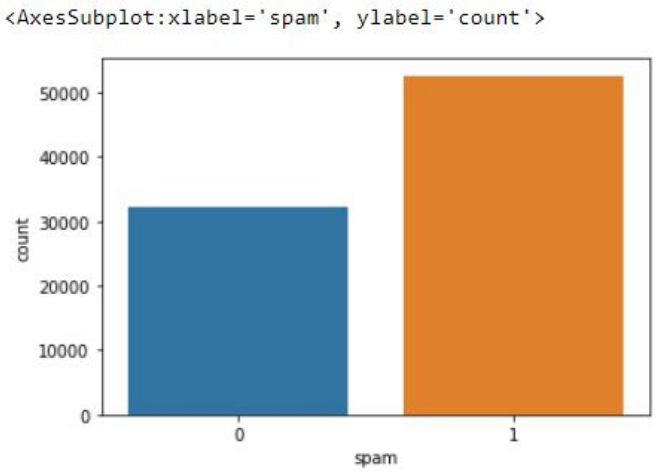

Abb. 2.1: Plot: Verteilung von Spam- bzw. Ham-Mails

spam	raw_data	subject	from	urls	domains	chars
1	Greetings!\n\nYou are receiving this letter be...	[ILUG] STOP THE MLM INSANITY	startnow2002@hotmail.com	[http://www.linux.ie]	[www.linux.ie]	3027
1	b'\n\n\n\n\n\n\n \n \n \n \n \n...	Life Insurance - Why Pay More?	12a1mailbot1@web.de	[http://website.e365.cc]	[website.e365.cc]	1423
1	\n\n\n\n\n\n\nThe Need For Safety Is Real In 200...	Real Protection, Stun Guns! Free Shipping! Ti...	lmrn@mailexcite.com	[http://www.geocities.com, http://www.geocitie...	[www.geocities.com, www.geocities.com, www.geo...	4349
1	1) Fight The Risk of Cancer!\nhttp://www.adcli...	[ILUG] Guaranteed to lose 10-12 lbs in 30 days...	taylor@s3.serveimage.com	[http://www.adclick.ws, http://www.adclick.ws,...	[www.adclick.ws, www.adclick.ws, www.adclick.w...	781
1	1) Fight The Risk of Cancer!\nhttp://www.adcli...	Guaranteed to lose 10-12 lbs in 30 days ...	sabrina@mx3.1premio.com	[http://www.adclick.ws, http://www.adclick.ws,...	[www.adclick.ws, www.adclick.ws, www.adclick.w...	636
...	...	...	...	...	...	...
0	Author: metze\nDate: 2007-04-16 07:41:12 +0000...	svn commit: samba r22248 - in\n\tbranches/SAMB...	metze@samba.org	[http://websvn.samba.org]	[websvn.samba.org]	840
0	On 4/9/07, Tom Phoenix wrote:\n>\n> On 4/9/07...	Re: OT: html checkbox question	jm5379@gmail.com	0	0	562
0	\n\nCharlie wrote:\n> Hi, this is Charlie and ...	Re: [R] Question for install Rapache package.	ligges@statistik.uni-dortmund.de	[https://stat.ethz.ch, http://www.R-project.or...	[stat.ethz.ch, www.R-project.org, stat.ethz.ch...	1178
0	Author: kseeger\nDate: 2007-04-16 07:47:27 +00...	svn commit: samba-docs r1098 - in trunk: manpa...	kseeger@samba.org	[http://websvn.samba.org, http://www.samba.org...	[websvn.samba.org, www.samba.org, www.samba.org]	12536
0	** ...	CBS SportsLine Daily Sports Report	mailer@mailer-relay.sportsline.com	[http://ww1.sportsline.com, http://ww1.sportsl...	[ww1.sportsline.com, ww1.sportsline.com, ww1.s...	8518

Abb. 2.2: Dataframe mit ungereinigten Zeichen

Clean Text

Wie in Abbildung 2.2 zu sehen ist, wurden die E-Mail Inhalte aus den Datensätzen zwar extrahiert (siehe Spalte „raw_data“), diese enthalten aber noch eine Vielzahl an Zeichen und Inhalte, die für ein weiteres Verarbeiten nicht gewollt sind. Deshalb wird in diesem Schritt der E-Mail Inhalt weiter aufbereitet und anschließend in einer neuen Spalte mit dem Namen „data“ gesichert.

Zur Aufbereitung wird eine Funktion mit dem Namen *clean_text* verwendet. Diese entfernt Folgendes:

- Steuerzeichen wie: \n, \r, \\n, \\r
- Alle URLs im Inhalt
- Alle E-Mail Adressen
- Alle nicht im Alphabet enthaltenen Zeichen, alles außer „a-z“ und „A-Z“

Im letzten Schritt wird der gesamte Inhalt in Kleinbuchstaben umgewandelt. Das Ergebnis dieser Operationen kann durch das Vergleichen der Spalten „raw_data“ und „data“ aus der nachfolgenden Abbildung 2.3 erkannt werden.

raw_data	subject	from	urls	domains	chars	data
Greetings!\n\nYou are receiving this letter be...	[ILUG] STOP THE MLM INSANITY	startnow2002@hotmail.com	[http://www.linux.ie]	[www.linux.ie]	3027	greetings you are receiving this letter becaus...
b'\n\n\n\n\n\n \n \n \n \n \n...	Life Insurance - Why Pay More?	12a1mailbot1@web.de	[http://website.e365.cc]	[website.e365.cc]	1423	b save up to on life insurance why spend more ...
\n\n\n\n\n\nThe Need For Safety Is Real In 200...	Real Protection, Stun Guns! Free Shipping! Ti...	lmrn@mailexcite.com	[http://www.geocities.com, http://www.geocitie...	[www.geocities.com, www.geocities.com, www.geo...	4349	the need for safety is real in you might only ...
1) Fight The Risk of Cancer!\nhttp://www.adcli...	[ILUG] Guaranteed to lose 10-12 lbs in 30 days...	taylor@s3.serveimage.com	[http://www.adclick.ws, http://www.adclick.ws,...	[www.adclick.ws, www.adclick.ws, www.adclick.w...	781	fight the risk of cancer p cfm o s pk slim dow...

Abb. 2.3: Dataframe mit gereinigten Zeichen

Tokenize Text

Unter Tokenization wird verstanden, dass ein Text in kleinere Bestandteile unterteilt wird. Diese Tokens können anschließend unter Anderem als Basis für Lemmatization verwendet werden [7].

Angewandt wurde hier Tokenization für einzelne Wörter. Folglich wird aus dem beispielhaften Inhalt „fight the risk of cancer“ anschließend „[fight, the, risk, of, cancer]“. Das Ergebnis wird in einer neuen Spalte „token_text“ im DataFrame abgelegt.

In einer weiteren Spalte wird die Anzahl der Tokens pro Eintrag gezählt und als „tokens“ an das DataFrame angehängt. Die beiden neuen Spalten werden auch in der Abbildung 2.4 abgebildet.

raw_data	subject	from	urls	domains	chars	data	token_text	tokens
ngs!\n\nYou are receiving this letter be...	[ILUG] STOP THE MLM INSANITY	startnow2002@hotmail.com	[http://www.linux.ie]	[www.linux.ie]	3027	greetings you are receiving this letter becaus...	[greetings, you, are, receiving, this, letter,...	510
b'\n\n\n\n\n\n \n \n \n \n \n...	Life Insurance - Why Pay More?	12a1mailbot1@web.de	[http://website.e365.cc]	[website.e365.cc]	1423	b save up to on life insurance why spend more ...	[b, save, up, to, on, life, insurance, why, sp...	173
\n\n\nThe Need For Safety Is Real In 200...	Real Protection, Stun Guns! Free Shipping! Ti...	lmrn@mailexcite.com	[http://www.geocities.com, http://www.geocitie...	[www.geocities.com, www.geocities.com, www.geo...	4349	the need for safety is real in you might only ...	[the, need, for, safety, is, real, in, you, mi...	533

Abb. 2.4: Dataframe mit angehängten Token-Informationen

Remove Stopwords

Stoppwörter sind in der jeweiligen Sprache häufig vorkommende Wörter wie „the“, „is“, „in“ oder „to“. Diese Wörter haben keine große Auswirkung auf den Informationswert eines Dokumentes und können deshalb ignoriert werden. Aufgrund dessen wird die Menge an Daten in einem Datensatz auf aussagekräftige Tokens reduziert [8].

In diesem Schritt werden aus „token_text“ alle Stoppwörter entfernt. Beispiele für diese können aus der nachfolgenden Liste entnommen werden:

‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’, ‘until’, ‘below’, ‘are’ [9]

Die von Stoppwörtern gereinigte Token-Liste wird nun als „stop_text“ an das DataFrame angeheftet, wie in der Abbildung 2.5 zu sehen ist. Bereits in der ersten Zeile kann im Vergleich zu „token_text“ erkannt werden, dass „stop_text“ keine Wörter wie „you“, „are“ oder „this“ mehr enthält.

raw_data	subject	from	urls	domains	chars	data	token_text	tokens	stop_text
re receiving this letter be...	[ILUG] STOP THE MLM INSANITY	startnow2002@hotmail.com	[http://www.linux.ie]	[www.linux.ie]	3027	greetings you are receiving this letter becaus...	[greetings, you, are, receiving, this, letter,...	510	[greetings, receiving, letter, expressed, inte...
\n\n \n \n \n \n \n...	Life Insurance - Why Pay More?	12a1mailbot1@web.de	[http://website.e365.cc]	[website.e365.cc]	1423	b save up to on life insurance why spend more ...	[b, save, up, to, on, life, insurance, why, sp...	173	[b, save, life, insurance, spend, life, quote,...
ed For Safety Is Real In 200...	Real Protection, Stun Guns! Free Shipping! Ti...	lmrn@mailexcite.com	[http://www.geocities.com, http://www.geocitie...	[www.geocities.com, www.geocities.com, www.geo...	4349	the need for safety is real in you might only ...	[the, need, for, safety, is, real, in, you, mi...	533	[need, safety, real, might, get, one, chance, ...

Abb. 2.5: Dataframe mit entfernten Stoppwörtern

Lemmatization

Lemmatization bildet die verschiedenen Ausprägungen eines Wortes zurück in seine Stammform ab. Ein Beispiel hierfür wären die Wörter „runs“, „running“ und „ran“. Diese besitzen alle dieselbe Stammform „run“, welche auch als „Lemma“ bezeichnet wird [10].

Nach der Lemmatization des „stop_text“ wurden alle Tokens in ihre Stammform überführt und in der neuen Spalte „clean_text“ abgelegt. Die Veränderung kann in der Abbildung 2.6 betrachtet werden. Beispielsweise hatte Lemmatization in der ersten Zeile folgende Auswirkungen „greetings“ -> „greet“, „receiving“ -> „receive“ oder „expressed“ -> „express“.

ɜw_data	subject	from	urls	domains	chars	data	token_text	tokens	stop_text	clean_text
his letter be...	[ILUG] STOP THE MLM INSANITY	startnow2002@hotmail.com	[http://www.linux.ie]	[www.linux.ie]	3027	greetings you are receiving this letter becaus...	[greetings, you, are, receiving, this, letter,...	510	[greetings, receiving, letter, expressed, inte...	[greet, receive, letter, express, interest, re...
ın \n \n...	Life Insurance - Why Pay More?	12a1mailbot1@web.de	[http://website.e365.cc]	[website.e365.cc]	1423	b save up to on life insurance why spend more ...	[b, save, up, to, on, life, insurance, why, sp...	173	[b, save, life, insurance, spend, life, quote,...	[b, save, life, insurance, spend, life, quote,...
/ Is Real In 200...	Real Protection, Stun Guns! Free Shipping! Ti...	lmrn@mailexcite.com	[http://www.geocities.com, http://www.geocitie...	[www.geocities.com, www.geocities.com, www.geo...	4349	the need for safety is real in you might only ...	[the, need, for, safety, is, real, in, you, mi...	533	[need, safety, real, might, get, one, chance, ...	[need, safety, real, might, get, one, chance, ...

Abb. 2.6: Dataframe mit lemmatisierten Wörtern

Remove Duplicates

Abschließend werden alle Einträge entfernt, die mehrfach im DataFrame vorhanden sind. Zur Identifikation wurde die Spalte „raw_data“ verwendet. Zusätzlich zu den Duplikaten werden alle Einträge mit weniger als fünf Tokens in „clean_text“ gelöscht. Diese besitzen kaum Inhalt, weshalb eine Textanalyse nicht sehr effektiv sein sollte. Die Ausführung dieser Operationen hat nachfolgendes Ergebnis:

All emails without duplicates	63570
Emails with less than five words	869
Emails with at least five words	62701

Tab. 2.1: Anzahl an Emails

2.1.3 Data Exploration

Outliers

Outliers sind Ausreißer innerhalb des Datasets und können einen negativen Einfluss auf das Training eines Modells haben, wenn es sich dabei um fehlerhafte Einträge handelt. Deshalb wurden die verschiedenen Einträge näher betrachtet. Bei Betrachtung der Zeichenanzahl wurden drei Ausreißer > 200.000 Zeichen gefunden, wie Abbildung 2.7 darstellt.

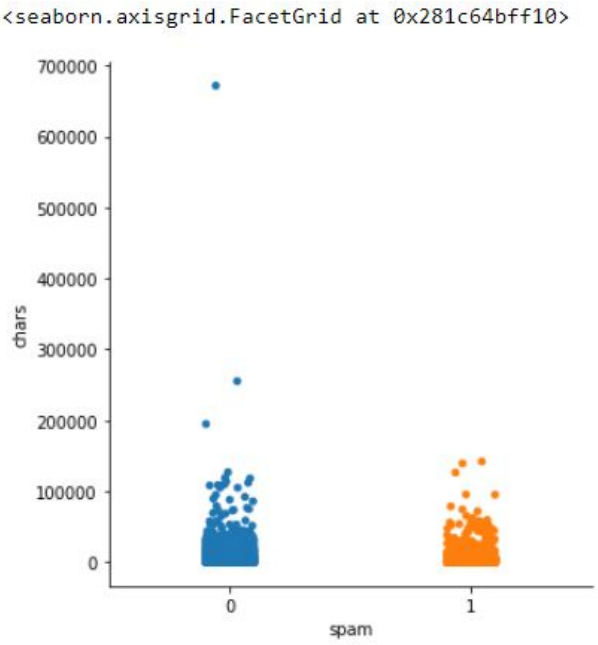

Abb. 2.7: 3 Ausreißer mit Zeichenanzahl > 200.000

In der nachfolgenden Abbildung 2.8 wurde versucht, diese drei Ausreißer herauszufiltern. Hierfür wurden alle Einträge mit über 190.000 Zeichen ausgewählt. Das Ergebnis sind drei E-Mails mit der Kennnummer 53324, 63715 und 75545.

```
In [6]: df[df['chars'].map(lambda d: d) > 190000]["raw_data"]

Out[6]: 53324    change your settings: http://blo.gs/settings.p...
        63715    On Fri Apr 27 14:43:55 2007, jkeen@verizon.net...
        75545    commit a snapshot of the generated prototype h...
        Name: raw_data, dtype: object
```

Abb. 2.8: Herausgefilterte Kennnummern der Ausreißer

Als nächstes wurden diese überprüft, um sicherzustellen, dass es sich bei ihnen nicht um fehlerhafte Einträge handelt. Der erste zu prüfende Eintrag hat die Kennnummer 63715 und ist in Abbildung 2.9 zu sehen. Dabei handelt es sich scheinbar um

eine E-Mail, in welcher Programmcode versandt wurde, was die hohe Anzahl an Zeichen erklärt. Ansonsten scheint die E-Mail aber fehlerfrei zu sein, weshalb sie nicht entfernt wurde.

```
df[df['chars'].map(lambda d: d) > 190000]["raw_data"][63715][:1000]
```

```
'On Fri Apr 27 14:43:55 2007, jkeen@verizon.net wrote:\n> A file with coverage analysis will be supplied in a  \n> separate pos
ting to this thread.\n\nAttached is a plain text version of the coverage analysis.  Coverage was run on a \'post-\nConfigure.pl
\' basis:\n\n  $ cover -delete coverage/configure/\n  $ ./myconfigure.sh \n  $ PERL5OPT=-MDevel::Cover=-db,coverage/configure/
prove t/configure/*.t t/\npostconfigure/01-data_slurp.t "$@"\n  $ cover coverage/configure/ -report=text  -ignore_re \'(^(confi
g\\/|lib\\/(?!P\narrot))|\\busr\\b)\' > ~/learn/parrot/coverage/postconfigure.coverage.txt\n\nkid51\nb\'Reading database from /
Users/jimk/work/fresh/coverage/configure\\n\\n\\n---------------------------- ------ ------ ------ ------ ------ ------ ------
\\nFile                          stmt   bran   cond    sub    pod   time  total\\n---------------------------- ------ ------ -
----- ------ ------ ------ ------\\n/usr/local/bin/prove           73.7   43.8    0.0   46.7    n/a    1.1   61.1\\nconfig/aut
o/aio.pm          '
```

Abb. 2.9: Outlier 63715

Selbiges gilt für die E-Mail mit der Kennnummer 75545. Auch hier scheint es sich um Programmcode zu handeln, wie in Abbildung 2.10 zu sehen ist.

```
df[df['tokens'].map(lambda d: d) > 90000]["raw_data"][75545][:1000]
```

```
'commit a snapshot of the generated prototype headers\n---\nsource/client/client_proto.h     |  366 ++\nsource/include/proto.h
|10128 ++++++++++++++++++++++++++++++ \n++++++++\nsource/nsswitch/winbindd_proto.h |  516 ++\nsource/utils/net_proto.h
|  382 ++\nsource/utils/ntlm_auth_proto.h   |   28 +\nsource/utils/passwd_proto.h      |   12 +\nsource/web/swat_proto.h
|   49 +\n7 files changed, 11481 insertions(+), 0 deletions(-)\ncreate mode 100644 source/client/client_proto.h\ncreate mode 10
0644 source/include/proto.h\ncreate mode 100644 source/nsswitch/winbindd_proto.h\ncreate mode 100644 source/utils/net_proto.h\n
create mode 100644 source/utils/ntlm_auth_proto.h\ncreate mode 100644 source/utils/passwd_proto.h\ncreate mode 100644 source/we
b/swat_proto.h\n\ndiff --git a/source/client/client_proto.h b/source/client/client_proto.h\nnew file mode 100644\nindex 000000
0..f5027e8\n--- /dev/null\n+++ b/source/client/client_proto.h\n@@ -0,0 +1,366 @@\n+#ifndef _CLIENT_PROTO_H_\n+#define _CLIENT_'
```

Abb. 2.10: Outlier 75545

Als nächstes wurde die Anzahl an Tokens pro E-Mail näher betrachtet, der entsprechende Plot kann in der folgenden Abbildung 2.11 betrachtet werden. Hier gibt es einen Eintrag der sehr stark ausschlägt, dieser wurde als nächstes betrachtet.

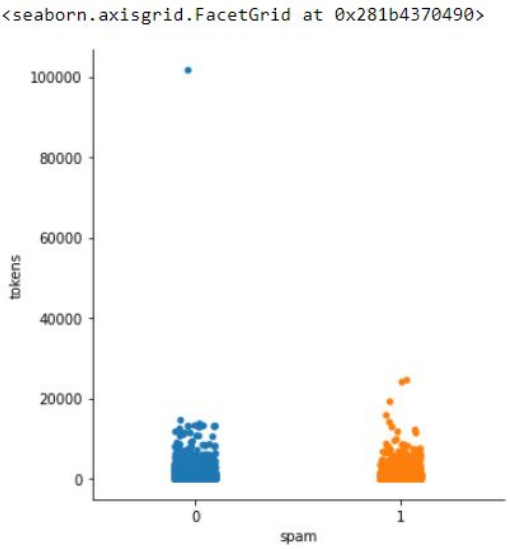

Abb. 2.11: 1 Ausreißer mit Tokenanzahl > 100.000

Dabei handelt es sich um den bereits betrachteten Eintrag mit der Kennnummer 75545, bei diesem handelt es sich wie bereits beschrieben um Programmcode.

Domain Blacklist Check (nicht implementiert)

Wie bereits beschrieben wurde, werden aus den E-Mails mehrere Elemente extrahiert, eines dieser sind Domains. Domains, die aufgrund von maliziösem oder dubiosem Netzwerkverkehr auffallen, werden häufig mit der Zeit in Domain-Blacklists gelistet. Deshalb wurde die Möglichkeit geprüft, in E-Mails enthaltene Domains durch eine Blacklist zu prüfen, um eine weitere Identifikation für Spam-E-Mails zu enthalten. Hierfür wurde die Blacklist `https://dbl.oisd.nl/` verwendet. Zu Beginn wurde für die Überprüfung einer Domain eine Suche in der Blacklist durchgeführt. Da dies jedoch eine schlechte Performance bei vielen Abfragen bedeutete, wurden alle Domains der Blacklist in eine Datenbank aufgenommen. Dies führte zu einer Verbesserung der Performance. Das Ergebnis der Prüfung des kompletten Datensatzes kann aus Abbildung 2.12 entnommen werden.

```
In [39]: df.groupby(['spam', 'bad_domain']).size()

Out[39]: spam  bad_domain
         0     0            31358
               1              813
         1     0            52453
               1              143
         dtype: int64
```

Abb. 2.12: Domain Check

Spam wird durch 1 repräsentiert, jedoch sind von über 50.000 Spam-E-Mails nur 143 durch einen Eintrag in der Blacklist betroffen. Bei den Ham-E-Mails sind es sogar 813, weshalb eine Domainprüfung leider keine aussagekräftige Identifikation für Spam-E-Mails darstellt. Der Grund hierfür könnte sein, dass eine tagessaktuelle Blockliste keine relevanten Einträge für Domains vor 20 Jahren mehr besitzt. Selbiges kann auch für Ham-E-Mails gelten - auch hier könnten Domains den Besitzer, Zielgruppe und Absicht gewechselt haben. Aufgrund der fehlenden Aussagekraft wurde deshalb die Idee der Domainprüfung gestrichen.

Feature Importance

Um nachvollziehen zu können, welche Features für die Entscheidung zwischen Spam und Ham besonders entscheidend sind, wurden diese für das trainierte Modell ausgegeben. Wie in dem in Abbildung 2.13 abgebildeten Graph zu sehen ist, sind einige der Features wesentlich wichtiger für die Entscheidungen des Modells als andere.

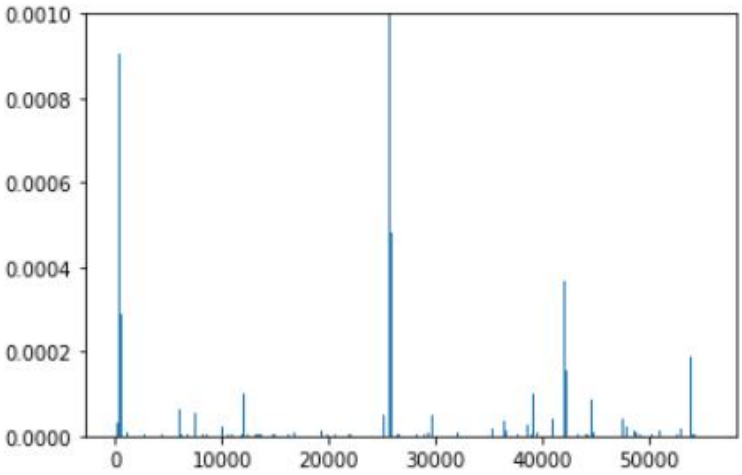

Abb. 2.13: Plot: Relevanz der Features auf Entscheidungen

Um zu verstehen welches Feature hinter den stärkeren Ausschlägen steht, wurden alle wichtigeren Features in der nachfolgenden Tabelle aufgelistet. Bei den beiden numerischen Einträge der Tabelle handelt es sich um die Anzahl der Tokens respektive die Anzahl der Zeichen.

summit	sidecar	cranford	asen	nerds
pullen	uac	ipo	polity	felix
patroclus	shelley	unconquered	incipient	caliber
momentarily	rashid	topics	55305	55306

Tab. 2.2: Tabelle mit wichtigen Features des Models

Wie am Ende der Tabelle zu sehen ist, sind die Angaben über die Anzahl an Tokens und Zeichen wichtige Faktoren für die Entscheidungen des Modells.

2.1.4 Vocabulary

Für das Training unseres Modells benötigen wir ein möglichst umfassendes Vocabulary, da dieses im Anschluss nicht mehr einfach geändert werden kann. Alle Wörter, die nicht im Vocabulary vorhanden sind, können später auch nicht zum Training eines Modells genutzt werden. Jedoch benötigt ein sehr umfassendes Vocabulary mehr Ressourcen, da es als X-Achse in einer Matrix eingesetzt wird.

Zu Beginn des Projektes war das Vocabulary so groß, dass nicht genug Arbeitsspeicher auf den verwendeten Rechnern zu Verfügung stand. Der Grund hierfür war, dass alle möglichen Wörter, die in den E-Mail Datensätzen vorhanden sind, als Vocabulary

eingesetzt wurden. Dieses beinhaltete jedoch invalide Wörter wie „aaaaaa“ oder „abc“, diese haben für eine Textanalyse keinen Wert, da sie keine legitimen Wörter in der englischen Sprache sind. Gelöst wird dieses Problem dadurch, dass ein komplettes Vocabulary mit allen Wörtern aus den Datensätzen erstellt und anschließend eine Schnittmenge mit externen Wortlisten durchgeführt wird. Auf diese Weise werden alle nicht legitimen Wörter aus dem Vocabulary entfernt. Zusätzlich bleiben legitime Wörter, die in den E-Mails vorkommen, enthalten. Das daraus resultierende Vocabulary ist kleiner als das ursprüngliche Vocabulary und kann deshalb vom Rechner verarbeitet werden. Weiterhin sind aber alle relevanten Wörter aus den Datensätzen vorhanden.

Die für die Schnittmenge verwendeten externen Vocabularys stammen aus GitHub und aus der NLTK-Bibliothek von Python [11]. Würden diese beiden nativ als Vocabulary eingesetzt werden, so entstünde wieder das ursprüngliche Problem und es würde an Ressourcen (im speziellen Arbeitsspeicher) fehlen. Wird ein kleines externes Vocabulary verwendet, besteht die Gefahr, dass legitime Wörter aus den Datensätzen nicht in diesem enthalten sind, weshalb diese auch nicht zum Training genutzt werden können.

2.1.5 Model

Prepare Dataset Parts

Initial wird das gesamte DataFrame in zwei DataFrames mit nur Spam und Ham getrennt. Diese werden dann individuell durch die „shuffle“ Funktion neu gemischt und anschließend die Zeilennummern wieder zurückgesetzt. Aus diesen beiden DataFrames werden anschließend jeweils 25.000 Ham und Spam Einträge für ein Trainings-DataFrame entnommen und dieses anschließend wieder neu durchgemischt. Aus den verbleibenden Einträgen in den Spam und Ham DataFrames werden für ein Test-DataFrame 3410 Exemplare verwendet und dort auch wieder gemischt.

Das Ergebnis dieser Vorbereitung ist, dass nun ein DataFrame mit insgesamt 50.000 E-Mail Einträgen mit einem Verhältnis von 50% Ham und Spam zum Training des Modells vorhanden ist. Es handelt sich hier um Downsampling, da wir nicht alle zur Verfügung stehenden Einträge für den Trainingsvorgang nutzen. Begründet ist das dadurch, dass Anzahl der zu Verfügung stehenden Spam- und Ham-E-Mails ungleich verteilt ist. Etwa 30.000 Ham-E-Mails stehen ca. 50.000 Spam-E-Mails gegenüber - und da noch einige Einträge für den abschließenden Test des Modells

gebraucht werden, wurden nur 25.000 Exemplare aus jeder Gruppe entnommen. Auf diese Weise bleibt der Trainings-DataFrame ausbalanciert. Das Ausbalancieren eines Datensatzes ist notwendig, da andernfalls das Model bzgl. der dominaten Klasse voreingenommen ist und sich häufiger für diese Klasse entscheiden wird [12]. Wie das Tranings-DataFrame enthält auch das Test-DataFrame eine ausgeglichene Anzahl an Spam und Ham Einträgen.

X-Wert und Y-Wert vorbereiten

Im ersten Schritt wird ein Corpus auf Grundlage von „clean_text“ aufgebaut. Dieser enthält nun für jeden E-Mail Eintrag den zugehörigen aufbereiteten Inhalt in Form des „clean_text“, welcher für TF-IDF benötigt wird.

TF-IDF kann für die Text-Analyse verwendet werden. Bei TF-IDF (term frequency-inverse document frequency) handelt es sich um ein statistisches Maß, welches zur Dokumentensuche und Informationsgewinnung entwickelt wurde. Zur Berechnung wird zum Einen die Häufigkeit von Wörtern in jeder E-Mail gezählt (term frequency), zum Anderen die Häufigkeit von Wörtern über alle E-Mail Inhalte (inverse document frequency). Ein Wort, das sehr häufig in den E-Mails vorhanden ist, wird mit einem Wert nahe 0 bewertet. Ist ein Wort in allen E-Mails selten vorhanden, wird ein Wert nahe 1 zugewiesen, denn diese haben eine höhere Relevanz für die Text-Analyse. TF-IDF wird benötigt, da vor der Verwendung der Texte zum Training eines Models diese noch in eine vektorisierte Form gebracht werden müssen. Deshalb wird auf Grundlage des Vocabularys die Bedeutung der Worte für jeden E-Mail Eintrag bestimmt [13].

Nach der Ausführung wird jeder E-Mail-Inhalt durch einen Vektor der Länge des Vocabularys dargestellt. Jeder Wert in diesem Vektor entspricht dabei dem TF-IDF-Score des jeweiligen Wortes.

An die jeweiligen Vektoren dieser Matrix wird anschließend die zugehörige Anzahl an Tokens und Zeichen angehängt. Damit repräsentiert jeder der Vektoren der Matrix den vektorisierten Inhalt, die Anzahl der Zeichen und die Anzahl der Tokens einer der E-Mails des Trainings-DataFrames. Diese Matrix wird nun als x-Wert für das Training des Modells verwendet. Nachfolgend ist in der Abbildung 2.14 die Matrix zu sehen. Jede Zeile repräsentiert eine E-Mail.

Der y-Wert ist ein Vektor, der für jeden E-Mail-Eintrag beschreibt, ob es sich dabei um Ham oder Spam handelt. Diese Information wird aus der „spam“ Spalte des DataFrame entnommen.

```
array([[   0.,    0.,    0., ...,    0.,   77.,  489.],
       [   0.,    0.,    0., ...,    0.,  161., 1528.],
       [   0.,    0.,    0., ...,    0.,   61.,  433.],
       ...,
       [   0.,    0.,    0., ...,    0.,  157., 1036.],
       [   0.,    0.,    0., ...,    0.,  238., 1682.],
       [   0.,    0.,    0., ...,    0.,  196., 1159.]])
```

Abb. 2.14: TF-IDF Matrix

Wahl des Trainings-Algorithmus mit Cross-Validation

Für das Training sowie die Cross-Validation des Modells wird die Python Bibliothek von scikit verwendet. Diese stellt verschiedene Funktionen bereit, um Modelle zu trainieren, zu testen (scoren) oder eine Cross-Validation durchzuführen [14].

Cross-Validation wird primär zur Fähigkeitsbewertung eines Models auf noch unbekannte Daten verwendet. Dabei wird eine begrenzte Stichprobe genutzt, um im Allgemein abzuschätzen, wie das Model auf Daten, die nicht während des Model-Trainings verwendet wurden, abschneiden wird. Ein Dataset wird dazu in mehrere Teile (k) unterteilt. Anschließend wird mit allen Teilen außer einem (k-1) ein Modell trainiert. Das ausgelassene Set an Daten wird nun für die Validierung des trainierten Modells genutzt. Dieser Vorgang wird nun mit einem anderen Part als Validierungsdatensatz wiederholt. Dies geschieht so lange, bis alle Teile einmal als Validierungsdaten verwendet wurden. Bei Aufteilung in fünf Parts finden demnach fünf Durchläufe statt [15].

Cross-Validation wurde genutzt, um mehrere Algorithmen für das Training eines Modells zu testen. Für jeden der Algorithmen wurde eine Cross-Validation für das beschriebene Dataset durchgeführt. Dieser Vorgang hatte das in Abbildung 2.15 zusehende Ergebnis für die drei Algorithmen: „Multinomial Naive Bayes“, „RandomForest“ und „K-Nearest Neighbors“

Neben der Genauigkeit des erzeugten Modells wurde zusätzlich die Zeit gemessen, die für den Cross-Validation Vorgang des jeweiligen Algorithmus benötigt wurde. Die höchste Genauigkeit besitzt der „Random Forest“ Algorithmus, der auch die längste Zeit in Anspruch genommen hat. Da die Genauigkeit mit ca. 98% etwa 10 Prozentpunkte höher ist als die des Zweitplatzierten „Multinomial Naive Bayes“ wurde entschieden, „Random Forest“ als präferierten Algorithmus einzusetzen. Aufgrund der hohen Genauigkeit, die durch die Cross-Validation ermittelt wurde, ist von einer Optimierung der Hyperparameter abgesehen worden.

```
Estimator: MultinomialNB()
Crossval-Duration: 1038.6484024524689 s
Mean accuracy: 0.8724744808166138

Estimator: RandomForestClassifier()
Crossval-Duration: 5761.310852527618 s
Mean accuracy: 0.9793206617388244

Estimator: KNeighborsClassifier()
Crossval-Duration: 2052.179085254669 s
Mean accuracy: 0.7200281590989088
```

Abb. 2.15: Ergebnisse der Cross-Validation diverser Algorithmen

Endgültiges Modell

Das endgültige Modell wurde mit dem „Random Forest“ Algorithmus und den gesamten Trainingsdaten trainiert. Der „Random Forest“ ist ein Algorithmus, welcher sich für Klassifikationsaufgaben im Maschinellen Lernen einsetzen lässt. In Falle des Projekts geht es um die Klassifikation von E-Mails in die beiden Kategorien Spam und Ham. Durch den Algorithmus werden verschiedene Entscheidungsbäume kombiniert, um eine bestmögliche Entscheidung zu treffen. Dabei gehört „Random Forest“ zu den sogenannten überwachten Lernmethoden - für den Lernvorgang werden also Daten benötigt, die bereits mit den gewünschten Labeln der Kategorien versehen wurden [16]. Der abschließende Test des Modells wird durch einen „score“ Befehl mit dem im Abschnitt “Prepare Dataset Parts“ vorab vorbereiteten Test-Dataset durchgeführt. Somit wurden dieses Mal alle Trainingsdaten nur für das Training und Testen mit den vorab vorbereiteten Testdaten vollzogen. Die daraus resultierende Genauigkeit kann in der nachfolgenden Abbildung 2.16 betrachtet werden und ist mit knapp 98% sehr hoch.

```
Estimator: RandomForestClassifier()
Train-Duration: 498.00077414512634 s
Accuracy: 0.9787390029325513
```

Abb. 2.16: Trainingsdauer und Genauigkeit des finalen Models

Dieses Modell wird nun im Anschluss exportiert, um es für andere Anwendungen zu nutzen. Außerdem muss nach einem Neustart des Jupyter Notebooks das Modell nicht mehr neu trainiert werden, es wird einfach in das Notebook geladen.

Modell Test mit „modernen“ E-Mails

Die zum Training genutzten E-Mail Datensätze sind bereits ca. 15 Jahre alt. Deshalb kann nicht ausgeschlossen werden, dass dies zu einer Verschlechterung der Vorhersage auf aktuelle E-Mails führen kann. Gründe dafür könnten sein, dass sich die Wortwahl von validem E-Mail-Verkehr oder die Herangehensweise von Spammenden über den Verlauf der Zeit verändert hat. Deshalb sollte die Fähigkeit des Modells, moderne E-Mails zuverlässig zu bewerten, durch einen abschließenden Testlauf näher untersucht werden.

Als Datengrundlage für diese Bewertung werden Spam und Ham E-Mails verwendet, die vorab von Freunden und Bekannten gesammelt wurden. Diese mussten zuvor aufbereitet werden. Beispielsweise wurde das Entfernen von Duplikaten und mehreren nicht englischsprachigen E-Mails durchgeführt. Aus diesen E-Mails wurde anschließend ein balancierter Testdatensatz erstellt. Das erzeugte Modell erzielte im ersten Durchlauf eine Genauigkeit von ca. 53%. Um dieses Ergebnis näher zu untersuchen, wurden die gesammelten E-Mails näher betrachtet. Im zu testenden Datensatz befanden sich zunächst viele sehr ähnliche Ham-E-Mails eines Marketing-Newsletters, sowie eines Newsletters über den Erhalt einer "Green Card". Es wird vermutet, dass diese E-Mails tendenziell nicht korrekt klassifiziert wurden. Aufgrund der hohen Anzahl und der ähnlichen Struktur konnte die Genauigkeit deshalb signifikant verschlechtert worden sein. Nach Verringerung der Anzahl dieser E-Mails im Testdatensatz wurde eine Erhöhung der Präzision auf 65% erzielt. Durch die Verwendung eines weiteren Datensatzes, welcher strukturell vielfältigere E-Mails enthielt, wurde eine Präzision von 76% erreicht. Generell wurden bei Tests eine sehr hohe Spanne der Scoring-Werte identifiziert. Dies ist möglicherweise damit zu begründen, dass die Anzahl der Ham-E-Mails deutlich höher waren als die der Spam-E-Mails. Durch randomisierte Auswahl für die Balancierung wurden deshalb womöglich manchmal mehr oder weniger verschiedene unterschiedliche E-Mails ausgewählt.

Für eine ausgiebige Evaluation müsste bestenfalls ein Datensatz aus vielen verschiedenen Spam- und Ham-E-Mails verwendet werden. Eine Möglichkeit, eine faire Bewertung des Modells vollziehen zu könnten, wäre beispielsweise durch die Nutzung eines standardisierten Test-Datensatzs.

2.2 Entwicklung des Webinterfaces

Nachdem im vorherigen Kapitel ein erfolgreicher Export der angelernten Daten stattgefunden hat, kann das fertige Modell nun verwendet werden, um eine Vorhersage über die Spam-Wahrscheinlichkeit von E-Mails zu treffen.

Bei der Diskussion bezüglich der praktischen Umsetzung unseres Spam-Moduls standen mehrere Fokuspunkte im Raum:

- *Plattformübergreifend*: Die Lösung sollte auf mehreren Plattformen (iOS, Windows, Linux) lauffähig sein
- *Flexibel- und erweiterbar*: Mithilfe eines modularen Programmieransatzes oder der Verwendung eines Schnittstellenmodels soll es möglich sein, andere Software an die bestehende Implementierung anzubinden.
- *Kompatibilität*: Die Softwarekomponenten müssen mit unserem exportiertem Modell zurechtkommen Dies würde auch Zeitersparnisse mit sich bringen, da keine neuen Bibliotheken für diverse Programmiersprachen gelernt werden müssen.

Unter Betrachtung dieser Punkte entschied sich die Gruppe, eine Web-basierte Software unter der Verwendung eines Python-Frameworks zu programmieren.

Damit können die bereits vorhandenen Kenntnisse der entsprechenden Machine-Learning-Bibliothekeb (scikit, NLTK etc.) direkt weiterverwendet werden. Zudem werden auch Komplikationen in Verbindung mit unserem exportiertem Modell verhindert. Aufgrund der Tatsache, dass die Gruppenmitglieder bereits Erfahrung in der Programmierung mit Flask aufgrund von vergangen Programmierprojekt haben, hat sich diese Lösung als Favorit herausgestellt.

2.2.1 Aufbau

Die Struktur der Applikation kann grob in ein Web bzw. HTML-Frontend, als auch das Backend eingeteilt werden. Der Quellcode und weitere Informationen können dem GitHub-Repository entnommen werden [17].

Frontend

Die Benutzeroberfläche der Anwendung besteht aus einem One-Pager. Es wird ein kurzer Überblick über die gebotene Funktionalität beschrieben und ein Upload-Feld, sowie ein Submit-Button abgebildet.

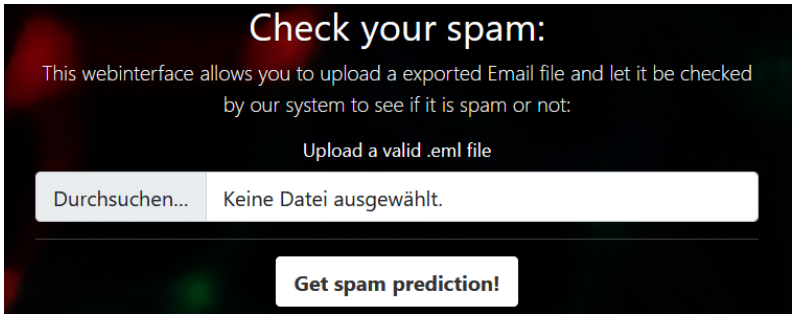

Abb. 2.17: Startseite der Webapplikation

Der Anwender kann eine exportierte Email-Datei (.eml-Format) im Hochladen-Fenster auswählen und anschließend den Button bestätigen. Er kriegt danach in einem Alert-Feld angezeigt, ob es sich bei seiner betroffenen Email um Spam (rot) oder um Ham (grün) handelt. In der Ausgabe befindet sich außerdem die Information, mit welcher Wahrscheinlichkeit das System diese Annahme trifft.

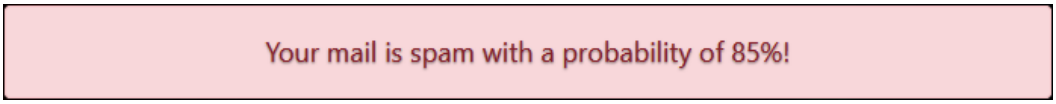

Abb. 2.18: Ausgabe bei einer Spam-Email

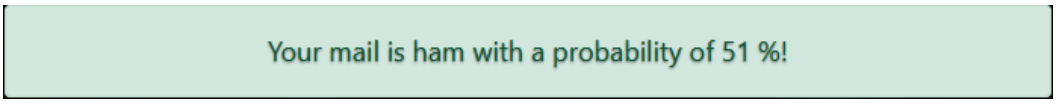

Abb. 2.19: Ausgabe bei einer Ham-Email

Nach dem erfolgreichen Analysieren einer Email kann der User erneut weitere Emails analysieren und den Prozess beliebig oft wiederholen.

Backend

Das Python Backend setzt zum großen Teil auch auf die selben Bibliotheken, welche bereits bei der Bearbeitung und Erstellung des initialen Modells zum Einsatz kamen, sowie einigen weiteren importierten Libraries zum Managen der ein- und ausgehenden HTTP-Requests, wenn auf die Webanwendung zugegriffen wird. Dank

der einfachen Möglichkeit, die Anwendung über einen einfachen HTTP-Request anzusprechen, können auch alternative Frontends oder Implementierungsmethoden verwendet werden, um eine Prediction von Emails zu ermöglichen, wie beispielsweise ein Kommandozeilen-Interfaces, welches direkt Input annehmen kann und auch ein Verarbeiten größerer Mengen an Emails ermöglicht.

Upload der Email

Das im Frontend implementierte Formular sendet einen HTTP-POST Request an die Webanwendung, welcher die Email-Datei als Multipart-Datei enthält. Dies findet über die „/predict“-Route der Webanwendung statt.

Wird stattdessen ein GET-Request verwendet (Browser-Aufruf), so wird die normale Prediction-Startseite aufgerufen.

Email-Handling

Im Anschluss findet ein am Server ein Processing der erhaltenen Email statt. Ähnlich dem Processing der Emails der initialen Datensätze wird die Email in diverse Teile aufgeteilt und in einem Dataframe gespeichert. Ebenso findet auch wieder ein Säubern und Tokenisieren statt (Entfernen von Stopp-Wörtern, Erkennen von Domains).

Sofern dies geschehen ist, kann die Predict-Funktion des Modells verwendet werden, um die Zuweisung der Email zur Spam bzw. Ham Klasse und die dazugehörige Wahrscheinlichkeit auszugeben und weiterzuverwenden.

Senden der Antwort.

Als Webframework besitzt Flask natürlich eine Funktion zum Erstellen einer passenden Antwort zum eingehenden HTTP-Request. Unter Verwendung eines HTML-Templates werden die eben zurückgegebenen Werte via Template-Variablen weitergegeben, und der Enduser erhält sein Ergebnis dargestellt.

2.2.2 Installation

Eine Bereitstellung der Web-Applikation ist auf Systemen jeglicher Plattform mit installiertem Python3.8+ möglich. Neben der Installation der benötigten Bibliotheken über den Python-Paketmanager pip, muss der Anwendung auch die Pfade zu den Vocabulary- und Model-Export-Dateien mitgeteilt werden. Dies geschieht in der Datei *predict_email.py*.

```
class Prediction:
  def __init__(self):
    vocab = joblib.load("./app/data/vocab.sav")
    self.model = joblib.load("./app/data/model.sav")
    self.tfidf = TfidfVectorizer(vocabulary=vocab)
```

Ein Start der Anwendung erfolgt mit dem Befehl *python3 run.py*. Es startet sich ein Webserver, welcher das Web-Frontend unter der Adresse 0.0.0.0:5000 verfügbar macht. Ein Testen von E-Mails ist ab diesem Zeitpunkt ab möglich. Für einen produktiven Einsatz empfiehlt sich das Vorschalten eines Reverse Proxys, welcher Sicherheitsmaßnahmen wie TLS, Rate-Limiting und Authentifizierung ergänzend umsetzen kann.

2.3 Entwicklung des Plugins

Zur Verbesserung der Bequemlichkeit bei der Verwendung des Models wurde nach weiteren möglichen Optionen gesucht. Möchten Nutzende eine E-Mail zur Prüfung an das Webinterface senden, so ist ein vorheriger Export dieser E-Mail als .eml-Datei notwendig. Je nach E-Mail-Programm (MUA - Mail User Agent) lässt sich dies aber unterschiedlich bewerkstelligen. Webbasierte Dienste, wie Microsoft Outlook Web App (OWA), bieten diese Funktion teilweise gar nicht.

Die Idee der Bereitstellung eines Kommandozeilentools (CLI - command line interface) zum automatischen Export und Upload der E-Mail (unter Nutzung des Webinterfaces) oder zur direkten Interaktion mit dem Model wurde verworfen. Nutzende müssten technische Kenntnisse zur Verwendung, insbesondere auf der Kommandozeile, mitbringen. Dies kann jedoch nicht immer gewährleistet werden.

Als Alternative Lösung wurde für das Mailprogramm “Thunderbird“ ein Plugin entwickelt, das eine einfache Integration mit dem Webinterface bietet. Thunderbird ist ein Open-Source Desktop-Mailprogramm der Mozilla Foundation [18]. Mithilfe der MailExtensions bietet das Programm dabei die Möglichkeit Add-ons durch gängige Webtechnologien (HTML, CSS, JS) zu entwickeln [19].

2.3.1 Installation

Die Erweiterung wird als .xpi-Datei bereitgestellt. Über das Add-ons-Menü von Thunderbird (siehe Abbildung 2.20) kann die Installation sowie die Konfiguration (URL

der Webinterface-Instanz) vorgenommen werden. Nähere Informationen können dem GitHub-Repository entnommen werden [20].

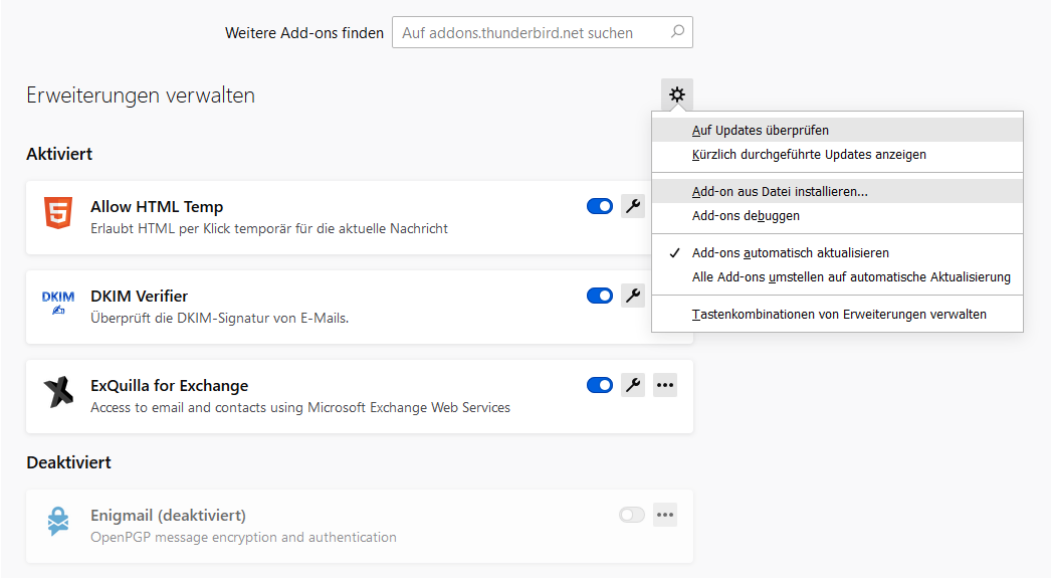

Abb. 2.20: Installation der Add-on-Datei

2.3.2 Nutzung

Die Erweiterung bietet zwei Möglichkeiten für die Verwendung. Eine E-Mail kann sowohl über die Nachrichtenübersicht ausgewählt als auch nach dem Öffnen über einen Schaltfläche (Button) geprüft werden. Ersteres ist vermutlich die bevorzugte Variante, da eine potentielle Spam-E-Mails aus Sicherheitsgründen ggf. gar nicht erst geöffnet werden möchte.

Bei Verwendung der Variante über die Nachrichtenübersicht geschieht dies mithilfe des Kontextmenüs (Rechtsklick) auf die ausgewählte E-Mail (siehe Abbildung 2.21). Es sei anzumerken, dass nicht mehrere Nachrichten gleichzeitig ausgewählt werden können.

Sofern der Endpunkt konfiguriert wurde und erreichbar ist, wird das Resultat über eine Desktop-Benachrichtigung, siehe Abbildung 2.22, bekanntgegeben. Falls dies nicht der Fall ist, wird eine entsprechende Fehlermeldung in der Benachrichtigung ausgegeben.

Vorausgesetzt, dass eine zu prüfende Nachricht bereits geöffnet wurde, bietet die Schaltfläche “Thunderham“ oben rechts eine Interaktion mit dem Webinterface. Das Resultat des Scans wird dabei unter der Schaltfläche als ein Tooltip, wie in

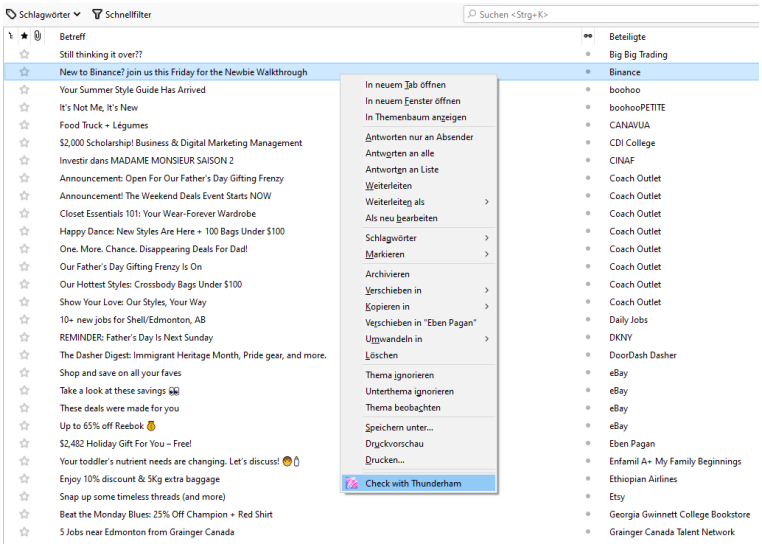

Abb. 2.21: Kontextmenüeintrag des Add-ons

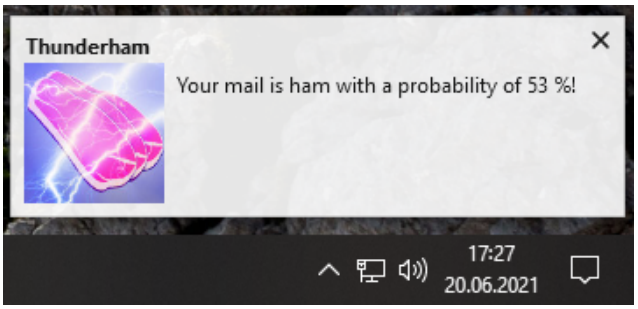

Abb. 2.22: Desktop-Benachrichtigung des Add-ons

der Abbildung 2.23 zu sehen ist, ausgegeben. Da es sich bei dem Tooltip um eine HTML-Seite handelt, können im Gegensatz zur Benachrichtigung, Farben verwendet werden. Damit wird Nutzenden ein positives (Ham - grün) oder negatives (Spam - rot) Ergebnis mittels Farben signalisiert.

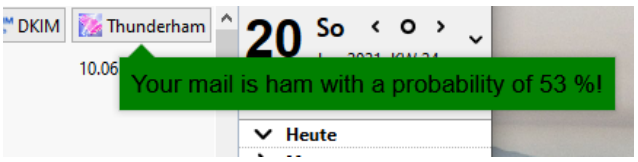

Abb. 2.23: Toolbar des Addons

2.3.3 Umsetzung

Das Addon als xpi-Datei ist ein unbenanntes Zip-Archiv, welches die Dateien für das Plugin enthält [21]. Eine Manifest-Datei enthält neben Metadaten wie den Namen

des Plugins oder eine dazugehörige Beschreibung auch Pfade zu den Icon-Dateien. Zudem werden Einstellungsmenü, das Popup-Menü (Button für geöffnete Nachricht) und im Hintergrund laufende Skripte definiert.

Das Hauptskript ‘background.js‘ erzeugt den Kontextmenüeintrag. Sofern eine E-Mail ausgewählt worden ist, wird die vollständige Nachricht (Headers und Body) an den Endpunkt des Webinterfaces mittels der Standard ‘fetch‘-API durch einen POST-Request übertragen. Die zu verwendende URL des Endpunkts wird aus einem Key-Value-Speicher extrahiert. Vorausgesetzt, dass die E-Mail erfolgreich an das Webinterface geschickt werden konnte, wird die HTML-Antwort, die das Resultat inkl. Prozentangaben für die Entscheidung enthält, mittels der ‘DOMParser‘-API ausgewertet. Die extrahierten Informationen werden dann mittels der Benachrichtigungsfunktion ausgegeben. Alternativ werden, z. B. bei einer undefinierten Serveradresse, Fehlermeldungen über die Benachrichtigungen ausgegeben.

Eine weitere JS-Datei liefert die Logik für das Popup-Menü. Der Ablauf für das Popup-Menü ist sehr ähnlich zu dem des Kontextmenüs. Anstelle der Erzeugung einer Benachrichtigung wird hier jedoch die HTML-Seite, die das Ergebnis inkl. Farben enthält, angepasst.

3 Fazit und Ausblick

Das Ziel dieser Arbeit war die Erkennung von Spam-E-Mails mithilfe eines Machine Learning Modells. Nutzenden können eigene englischsprachige E-Mails als .eml-Datei über ein Webinterface hochladen und eine Bewertung dazu erhalten. Zudem wird durch ein E-Mail-Programm-Plugin eine weitere Interaktionsmöglichkeit mit dem Modell, über das Webinterface, zur Verfügung gestellt.

Bei der Klassifikation von Spam und Ham E-Mails fokussiert sich das Modell ausschließlich auf die sprachliche Zusammensetzung dieser. Dabei wird der Kontext von E-Mails für die Entscheidung nicht berücksichtigt. Daher können klassische Spam-Emails, welche Themen wie Gewinnspiele, Potenzmittel oder Gratisgewinne behandeln, zuverlässig klassifiziert werden. Jedoch fällt es dem Model bei Newsletter, z. B. zur Produktangeboten oder Marketing-Kursen, oder Spear-Phishing schwer, diese korrekt zuzuordnen.

Eine Verbesserung des Modells wäre durch Vergrößerung und Ausweitung des Datensatzes und Vokabulars möglich. Darüber hinaus könnte mit einem größeren Testdatensatz eine bessere und praxisnähere Bewertung des Modells durchgeführt werden. Auch eine Erkennung schadhafter E-Mails wäre realisierbar, wenn eine Betrachtung der E-Mail-Anhänge vorgenommen werden würde. Erweitert um eine kontextuelle Erkennung von Inhalten, könnte so auf Basis unseres Models eine ganzheitliche Lösung zur Erkennung von Spam-Emails erstellt werden.

Der Programmcode sowie weitergehende Informationen zu den entwickelten Lösungen können dem GitHub Repository `https://github.com/konstantingoretzki/spamdetection` entnommen werden.

Quellenverzeichnis

[1] Bundesamt für Sicherheit in der Informationstechnik. (). “Spam”, Adresse: https://www.bsi.bund.de/SharedDocs/Glossareintraege/DE/S/Spam.html (besucht am 24. 06. 2021).

[2] Apache Foundation. (). “Index of /old/publiccorpus”, Adresse: https://spamassassin.apache.org/old/publiccorpus/ (besucht am 24. 06. 2021).

[3] ——, (). “REVISION HISTORY OF THIS CORPUS”, Adresse: https://spamassassin.apache.org/old/publiccorpus/readme.html (besucht am 24. 06. 2021).

[4] Cormack, Gordon; Lynam, Tom. (). “2007 TREC Public Spam Corpus”, Adresse: https://plg.uwaterloo.ca/~gvcormac/treccorpus07/ (besucht am 24. 06. 2021).

[5] ——, (). “TREC 2007 Public Corpus”, Adresse: https://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html (besucht am 24. 06. 2021).

[6] Microsoft. (). “A description of the multipart/mixed Internet message format”, Adresse: https://docs.microsoft.com/en-us/exchange/troubleshoot/administration/multipart-mixed-mime-message-format (besucht am 24. 06. 2021).

[7] Guru99. (). “NLTK Tokenize: Words and Sentences Tokenizer with Example”, Adresse: https://www.guru99.com/tokenize-words-sentences-nltk.html (besucht am 24. 06. 2021).

[8] Singh, Shubham. (). “NLP Essentials: Removing Stopwords and Performing Text Normalization using NLTK and spaCy in Python”, Adresse: https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/ (besucht am 24. 06. 2021).

[9] GeeksforGeeks. (). “Removing stop words with NLTK in Python”, Adresse: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/ (besucht am 24. 06. 2021).

[10] Jabeen, Hafsa. (). “Stemming and Lemmatization in Python”, Adresse: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python (besucht am 24. 06. 2021).

[11] dwyl. (). “List Of English Words”, Adresse: https://github.com/dwyl/english-words (besucht am 24. 06. 2021).

[12] Nagidi, Jaiganesh. (). “Best Ways To Handle Imbalanced Data In Machine Learning”, Adresse: https://dataaspirant.com/handle-imbalanced-data-machine-learning/ (besucht am 24. 06. 2021).

[13] Stecanella, Bruno. (). “What is TF-IDF?”, Adresse: https://monkeylearn.com/blog/what-is-tf-idf/ (besucht am 24. 06. 2021).

[14] scikit-learn. (). “Cross-validation: evaluating estimator performance”, Adresse: https://scikit-learn.org/stable/modules/cross_validation.html (besucht am 24. 06. 2021).

[15] Brownlee, Jason. (). “A Gentle Introduction to k-fold Cross-Validation”, Adresse: https://machinelearningmastery.com/k-fold-cross-validation/ (besucht am 24. 06. 2021).

[16] Luber, Stefan; Litzel, Nico. (). “Was ist Random Forest?”, Adresse: https://www.bigdata-insider.de/was-ist-random-forest-a-913937/ (besucht am 24. 06. 2021).

[17] Konstantin Goretzki. (). “spamdetection-ui”, Adresse: https://github.com/konstantingoretzki/spamdetection-ui (besucht am 24. 06. 2021).

[18] Mozilla Foundation. (). “Mozilla Thunderbird”, Adresse: https://www.thunderbird.net/de/ (besucht am 24. 06. 2021).

[19] ——, (). “A Guide to MailExtensions”, Adresse: https://developer.thunderbird.net/add-ons/mailextensions (besucht am 24. 06. 2021).

[20] Konstantin Goretzki. (). “spamdetection-plugin”, Adresse: https://github.com/konstantingoretzki/spamdetection-plugin (besucht am 24. 06. 2021).

[21] Mozilla Foundation. (). “Hello World Example”, Adresse: https://developer.thunderbird.net/add-ons/mailextensions/hello-world-add-ons (besucht am 24. 06. 2021).

Abbildungsverzeichnis