**KI-Projekt 1**

**Daten:**

SMS Spam Collection v.1:

Beschreibung: Der Datensatz besteht aus 5.574 SMS, die für die Forschen von SMS-Spam gesammelt wurden. „Ham“ steht für eine echte SMS und „Spam“ für eine Spam-SMS.

Diese 5.574 SMS wurden aus verschiedenen Quellen von Tiago Agostino de Almeida und José María Gómez Hidalgo zusammengetragen.

Quellen die für SMS Spam Collection v1 benutzt wurden:

* 425 Spam-SMS von der Website grumbletext [[1]](#footnote-1)
* 450 SMS von Caroline Tag (PhD Theses) [[2]](#footnote-2)
* 3.375 SMS von der Sammlung: NUS SMS (NSC) [[3]](#footnote-3)
* 1.002 SMS von dem Datensatz SMS Spam Corpus v0.1 [[4]](#footnote-4)
* 322 Spam-SMS von dem Datensatz SMS Spam Corpus v0.1 [[5]](#footnote-5)

Das Heist der Datensatz SMS Spam Collection v.1 hat insgesamt 4.827 SMS und 747 Spam-SMS

Features:

Hinzugefügt:

* NumChars -> die Anzahl der Buchstaben in der E-Mail.
* containsNumSequence -> ob Zahlen vorhanden sind. (Mindestens fünf Zahlen hintereinander)
* containsCurrencySymbol -> ob ein Währungszeichen vorhanden ist.
* numBuzzwords -> ob die Nachricht gewisse „Spam Wörter“ enthält. (zum Beispiel: Call „free“, „txt“, „prize“ usw.)
* percentUppercaseLetters -> Wie viel Prozent der Nachricht in Großbuchstaben geschrieben wurde.
* containsURL -> ob eine URL in der Nachricht ist.
* numExclamationMark -> Anzahl der Ausrufezeichen. („!“).
* numAbbreviations -> Anzahl der Abkürzungen in der Nachricht.

Schon vorhanden:

* class (ham, spam) -> **Ziel-Klasse**. **Sagt aus, ob es eine Spam-Nachricht ist oder nicht.**
* Text -> Die Nachricht in Textformat, die entweder ham oder spam ist.

**Trainingsprozess**

Der rohe Datensatz hat insgesamt 5574 Instanzen, 2 vorgegebene Attribute und 8 eigens hinzugefügte Attribute. Des Weiteren hat ein Testdatensatz 557 Instanzen, ein Trainingsdatensatz hat 4516 Instanzen und ein Entwicklungsdatensatz hat 451 Instanzen.

Erster Test:

Der erste Test wurde von allen Mitgliedern der Gruppe gemeinsam im Discord call gemacht. Vorgegangen sind wir wie in der Vorlesung. Wir haben einen kleinen Teil (10%) von dem Datensatz abgespalten[[6]](#footnote-6) und dann die Werte aufgezeichnet. Dafür haben wir die SMSSpamCollection6\_DEF.arff[[7]](#footnote-7) erstellt. Der Test hat 557 Instanzen und 6 Attribute.

Die wichtigen Werte, die wir gesammelt haben, sind:

=== Detailed Accuracy By Class ===

TP Rate | FP Rate | Precision | Recall | F-Measure | MCC | ROC Area | PRC Area | Class

0, 992 | 0,060 | 0,989 | 0,992 | 0,990 | 0,937 | 0,976 | 0,992 | ham

0,940 | 0,008 | 0,952 | 0,940 | 0,946 | 0,937 | 0,976 | 0,934 | spam

Weig. Avg. 0,984 | 0,052 | 0,984 | 0,984 | 0,984 | 0,937 | 0,976 | 0,983

=== Confusion Matrix ===

a b <-- classified as

469 4 | a = ham

5 79 | b = spam

Nach dem Test haben wir besprochen, dass es unser Ziel ist die Anzahl der (Ham) Nachrichten die als (Spam) erkannt wurden zu minimieren, da wir es wichtiger finden das alle (Ham) Nachrichten auch wirklich zugestellt werden. (es ist weniger schlimm, wenn ein paar Spam Nachrichten durch kommen.)

Zweiter Test:

Den zweiten Test haben wir im Unterricht durchgeführt. Dafür haben wir SMSSpamCollection\_complete\_DEF.arff [[8]](#footnote-8)erstellt, welcher ebenfalls 557 Instanzen und diesmal 9 Attribute hatte.

Die wichtigen Werte, die wir gesammelt haben, sind:

=== Detailed Accuracy By Class ===

TP Rate | FP Rate | Precision | Recall | F-Measure | MCC | ROC Area | PRC Area | Class

0, 996 | 0,036 | 0,994 | 0,996 | 0,995 | 0,965 | 0,984 | 0,995 | ham

0,964 | 0,004 | 0,976 | 0,964 | 0,970 | 0,965 | 0,984 | 0,945 | spam

Weig. Avg. 0,991 | 0,031 | 0,991 | 0,991 | 0,991 | 0,965 | 0,984 | 0,987

=== Confusion Matrix ===

a b <-- classified as

471 2 | a = ham

3 81 | b = spam

Diese Daten sind deutlich besser und es wurden auch gerade mal 5 Nachrichten falsch erkannt. Nach wie vor ist es immer noch unser Ziel, die falsch erkannten Ham-Nachrichten so weit wie möglich zu verringern.

Dritter Test:

Beim dritten Test wird getestet, ob gewisse Attribute überhaupt einen guten Einfluss auf die Testergebnisse haben. Für den Test wurde SMSSpamCollection\_complete\_DEF-2.arff [[9]](#footnote-9)erstellt, welcher 557 Instanzen und 8 Attribute hat. Für den Test wurde das Attribut „numBuzzwords“ entfernt, welches einige Spam-Wörter erkennen soll.

Außerdem hoffen wir durch das Weglassen von Attributen, die Spam erkennen, dass weniger False Negative erkannt werden.

Die wichtigen Werte, die wir gesammelt haben, sind:

=== Detailed Accuracy By Class ===

TP Rate | FP Rate | Precision | Recall | F-Measure | MCC | ROC Area | PRC Area | Class

0,996 | 0,060 | 0,989 | 0,996 | 0,993 | 0,950 | 0,977 | 0,992 | ham

0,940 | 0,004 | 0,975 | 0,940 | 0,958 | 0,950 | 0,977 | 0,947 | spam

Weig. Avg. 0,987 | 0,051 | 0,987 | 0,987 | 0,987 | 0,950 | 0,977 | 0,985

=== Confusion Matrix ===

a b <-- classified as

471 2 | a = ham

5 79 | b = spam

Im Vergleich zum zweiten Test, welcher 3 falsch erkannte Spam-Nachrichten aufweist, sind sie hier auf 5 erhöht worden. Diese Änderung ist jedoch gering, so dass es keine allzu große Veränderung bewirkt.

Vergleiche Endwerte mit dem 2.Test:

* Gelb = leicht schlechter
* Grün = unverändert
* Cyan = insgesamt schlechter

Als Ergebnis kann man feststellen, dass unsere Spam-Wörter zwar hilfreich sind bei der Erkennung von Spam-Nachrichten, aber keine gravierenden Auswirkungen bei unseren korrekt erkannten Ham-Nachrichten hat.

**Projektdokumentation**

Aufteilung der Arbeit:

* Aaron -> Daten, zusatzpunkte, Dokumentation
* Lisa -> Trainingsprozess, Dokumentation
* Patrick -> Merkmalsauswahl

Nach dem ersten Test:

Nachdem wir den ersten Test so wie oben in Trainingsprozess beschrieben durchgeführt haben, herausgefunden haben, wo unsere KI noch Verbesserungsmöglichkeiten hat, haben wir uns das weitere Vorgehen mit diesem Projekt überlegt. Wir sind zu dem Entschluss gekommen, dass wir durch das hinzufügen oder weglassen von gewissen Merkmalen eventuell zu einem besseren Ergebnis kommen können. Patrick hat daraufhin weitere Merkmale hinzugefügt und SMSSpamCollection\_complete.arff [[10]](#footnote-10) erstellt.

Nach dem zweiten Test:

Zwischen dem ersten und dem zweiten Test haben wir 3 weitere Attribute hinzugefügt. Nach dem wir den Test durchgeführt haben, haben wir gemerkt, dass dieser deutlich besser ausgefallen ist. Demzufolge arbeiten wir in die richtige Richtung.

Nach dem dritten Test:

Nachdem dritten Test und der Auswertung der Daten haben wir gemerkt das durch das Weglassen von 1-2 weiteren Attributen eventuell unser Ziel (keine False Negative) erreicht werden kann.

**Abschlussevaluation**

Die Performanz des Lerners bezüglich der Testdaten ist bereits fast perfekt. Anhand der Daten von Test-2 kann man jedoch erkennen, dass der Lerner leider mehr False Negative hat als False Positive. Dies führt dazu, dass Ham Nachrichten nicht als Ham anerkannt werden. Daraus lässt sich schließen, dass wir den Lerner nicht als Einsatztauglich ansehen. Hätten wir mehr False Positive aber dafür keine False negative, dann hätten wir das Ziel erreicht und den Lerner als einsatztauglich angesehen.

Eine Sache die wir aus den Datensätzen und den ersten Test bemerkt haben, ist dass der Lerner Bereits mit wenigen Attributen ein relatives gutes Ergebnis erzielt, es ist jedoch sehr schwer ist 100 Prozent Precision zu erreichen. Aufgrund der Eigenschaften von Spam Nachrichten, die oft gleich und immer strukturiert sind, kann man die Anzahl der False Negative eventuell auf null reduzieren durch das Weglassen von Attributen, was dann jedoch auch zu mehr false Positive führt.

**Zusatzpunkte:**

Eine weitere Sache, die wir versucht haben, ist die Verwendung eines anderen Lerners für den Datensatz. Benutzt haben wir dabei den Naive-Bayes. Dieser hat jedoch in Tests, egal ob viele oder wenige Attribute, nicht so gut funktioniert wie der Entscheidungsbaum. Der Naive-Bayes arbeitet anders als der Entscheidungsbaum. Er berechnet Wahrscheinlichkeiten anhand von Testdaten. Er geht außerdem davon aus, dass alle Attribute unabhängig voneinander sind. Eventuell war das bei unseren Attributen nicht der Fall, was zu einem schlechteren Ergebnis führen kann. Ein gutes Beispiel für den Naive-Bayes findet man in dem Buch: Grundlagen der Künstlichen Intelligenz Eine nichttechnische Einführung[[11]](#footnote-11) von Tom Taulli in Kapitel 3.

1. http://www.grumbletext.co.uk/ [↑](#footnote-ref-1)
2. http://etheses.bham.ac.uk/253/1/Tagg09PhD.pdf [↑](#footnote-ref-2)
3. http://www.comp.nus.edu.sg/~rpnlpir/ [↑](#footnote-ref-3)
4. http://www.esp.uem.es/jmgomez/smsspamcorpus/ [↑](#footnote-ref-4)
5. http://www.esp.uem.es/jmgomez/smsspamcorpus/ [↑](#footnote-ref-5)
6. https://github.com/21meaa/KI\_Projekt1/blob/main/Merkmalsauswahl/SMSSpamCollection6\_DEF.arff [↑](#footnote-ref-6)
7. https://github.com/21meaa/KI\_Projekt1/blob/main/Merkmalsauswahl/SMSSpamCollection6.arff [↑](#footnote-ref-7)
8. https://github.com/21meaa/KI\_Projekt1/blob/main/Merkmalsauswahl/SMSSpamCollection\_complete.arff [↑](#footnote-ref-8)
9. https://github.com/21meaa/KI\_Projekt1/blob/main/Merkmalsauswahl/SMSSpamCollection\_complete\_DEF-2.arff [↑](#footnote-ref-9)
10. https://github.com/21meaa/KI\_Projekt1/blob/main/Merkmalsauswahl/SMSSpamCollection\_complete.arff [↑](#footnote-ref-10)
11. [Grundlagen der Künstlichen Intelligenz: Eine nichttechnische Einführung | SpringerLink](https://link.springer.com/book/10.1007/978-3-662-66283-0) [↑](#footnote-ref-11)