Sarkasmuserkennung

Im Bereich des Natural Language Processing (NLP) gibt es verschiedene Anwendungsziele. Viele beschäftigen sich mit dem Inhalt des Gesprochenen oder des Textes. Ein Fall, der nicht nur für den Computer eine Herausforderung bietet, sondern auch häufiger für den Menschen selbst, ist das Erkennen von Sarkasmus. Der Duden beschreibt Sarkasmus als: "beißender, verletzender Spott, Hohn, der jemanden, etwas lächerlich machen will". Die Schwierigkeit ist meist, herauszufinden, ob etwas ernst oder sarkastisch gemeint ist. Es benötigt eine genaue Analyse des Kontextes. Oftmals sind es auch nur Betonungen, die verraten, dass es sich um Sarkasmus handelt. In direkter Verbindung dazu steht die Ironie, die immer das Gegenteil der getätigten Aussage bezeichent. Ironie ist nicht immer auch sarkastisch, kann aber im Zuge dessen verwendet werden.

Um textuellen Sarkasmus zu erkennen, muss der Text vor allem inhaltlich klassifiziert werden. Dies wird mit gängigen Methoden des Natural Language Processings und Text Mining realisiert.

Ziel des Projektes ist es, ein Model zu entwickeln, welches anhand eines Textes zurückliefert, ob es sich um Sarkasmus handelt oder nicht.

Projektbeschreibung

In diesem Projekt soll versucht werden mittels künstlicher Intelligenz und Methoden des Text-Mining Sarkasmus zu erkennen. Dafür wurde ein Datansatz ausgesucht, der aus Überschriften von Onlineartikeln besteht, welche aus zwei unterschiedlichen Quellen stammt. Die eine ist eine Satire-Seite ("The Onion"), die andere ist eine seriöse Pressestelle ("The Huffington Post"). Die Überschriften sind gelabelt und sollen Klassifiziert werden.

Datenbeschaffung

Die Daten wurden auf Kaggle bereitgestellt und werden im Folgenden eingelesen. Ein Datensatz besteht aus drei Komponenten: die zu klassifizierende Überschrift, einem Label und einem Link zu dem entsprechenden Artikel. Das Label gibt an, ob ein Artikel sarkastisch ist (1) oder nicht (0).

Web-Crawler

Um genauere Klassifizierungen zu gewährleisten, wird zu jedem Datensatz der Inhalt des Artikels benötigt, so dass in der Klassifizierung ein Kontext zu der Überschrift existiert. Der Inhalt des Artikels wird aus dem Link im Datensatz ausgelesen. Existiert kein Artikel mehr oder gibt es zu einer Überschrift keinen Text, sondern nur ein Bild, wird dieser Datensatz aus dem Gesamtsatz gestrichen.

CODE ANZEIGEN

Crawler Huffpost

```
<body>
  <div class="body-wrap">
    <div class="main">
      <div id="main">
        <div class="entry__body js-entry-content">
          <div class="entry__body js-entry-body">
            <div class="entry__text js-entry-text yr-entry-text">
              <div class="content-list-component yr-content-list-text text">
                <a></a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
```

Dieser HTML-Text stellt die Verschachtelung bis zum Inhalt des Artikels dar.

CODE ANZEIGEN

Crawler Onion

```
<body>
  <div>
    <div>
      <main>
        <div>
```

```
          <div class="js_starterpost">
            <div class="r[...]lxo-0 hEDDLA js_post-content">
              <p></p>
            </div>
          </div>
        </div>
      </main>
    </div>
  </div>
</body>
```

Dieser HTML-Text stellt die Verschachtelung bis zum Inhalt des Artikels dar.

CODE ANZEIGEN

Daten einlesen

CODE ANZEIGEN

CODE ANZEIGEN

Daten bereiningen

Um mit den beschafften Daten arbeiten zu können, müssen diese zuvor bereinigt werden. Das bedeutet, dass mit Daten umgegangen werden muss, die unvollständig oder falsch sind. Im Falle dieses Projektes werden die Daten gelöscht.

CODE ANZEIGEN

Außerdem können Werte, die nicht gebraucht werden, gelöscht werden. In diesem Fall handelt es sich um die Spalte "article_link". Sie ist nach dem Auslesen des Kontextes nicht mehr von Nutzen.

CODE ANZEIGEN

CODE ANZEIGEN

EDA und Feature-Extraktion

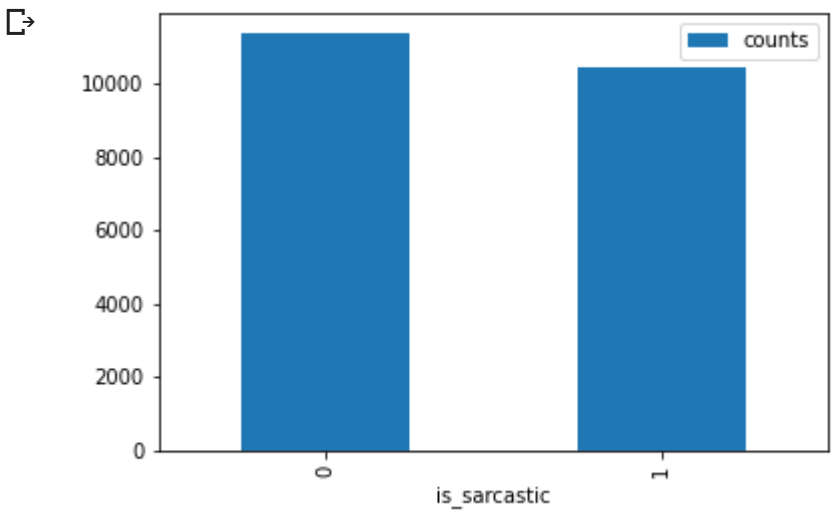
Um Sarkasmus in Texten zu erkennen, muss der Text analysiert werden und die Features zur Klassifizierung extrahiert werden.

Komponente	*Features*
Überschrift	• Sentiment • Emotionen • Slang
Inhalt	• Sentiment • Emotionen • Slang

Verteilung der Daten

Ein Blick auf die Verteilung der Daten gibt einen Überblick darüber, ob die Balance stimmt. Gibt es viel mehr sarkastisch gelabelte Daten als nicht sarkastische, könnte das das Training des Neuronalen Netzes negativ beeinflussen, da ein Ungleichgewicht an Informationen herrscht. Um sicherzustellen, dass das Training für sarkastisch und nicht sarkastisch gleichermaßen gut funktioniert, muss dafür gesorgt werden, dass im Datensatz ungefähr gleich viele gelabelte Daten von beiden Kategorien vorhanden sind. Insgesamt besteht der Datensatz aus 21799 gelabelten Artikeln.

CODE ANZEIGEN



Die Zählung der Elemente aus dem Datensatz ergab einen geringen Überschuss an nicht sarkastischen Daten. Es ist aber auch eine Balance zu sehen, daher können die Daten ohne weitere Maßnahmen zum Trainieren verwendet werden.

Wordcloud

Für einen Überblick über die Daten und ihren Inhalt lässt sich eine Wordcloud verwenden. Sie beschreibt die am häufigsten vorkommenden Worte in den gegebenen Texten. Im Folgenden wird unterschieden zwischen sarkastischen Texten und nicht sarkastischen Texten.

CODE ANZEIGEN

CODE ANZEIGEN

```
Wordcloud für die sarkastischen Headlines:
```


```
Wordcloud für die nicht sarkastischen Headlines:
```

Wort- und Textlänge

Ein weiteres Feature im NLP, das zur Analyse eines Textes verwendet werden kann, ist die Anzahl der Worte, also die Länge des Textes.

In seriösen Artikeln soll der Leser sachlich informiert werden. Wir vermuten, dass sich dieses in der Textlänge wiederspiegeln könnte, da in einem objektiven Artikel der Sachverhalt aus mehreren Perspektiven betrachtet wird, während das bei sarkastischen Texten nicht der Fall sein muss, da dieser durchaus polarisierend geschrieben sein kann.

Zudem ist ein objektiver, sachlicher Artikel eines seriösen Nachrichtenportals nicht zwangsläufig in einfacher Sprache geschrieben, was sich in der durchschnittlichen Wortlänge wiederspiegeln könnte und gleichzeitig auch in einem längeren Text.

CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

Textlänge anhand von Anzahl an Worten:

CODE ANZEIGEN

```
Durchnittliche Anzahl an Worten in nicht sarkastischen Überschriften:  9.85
Durchnittliche Anzahl an Worten in sarkastischen Überschriften:  10.62
Durchnittliche Anzahl an Worten in nicht sarkastischen Artikeln:  13252.12
Durchnittliche Anzahl an Worten in sarkastischen Artikeln:  151.22
```

Beispiel anhand zufällig ausgewählte Artikel:

CODE ANZEIGEN

```
Sarkastischer Artikel:  1
shadow government getting too large to meet in marriott conference room b
COLUMBUS, OH—With its membership swelling in recent months, the mysterious or
```

CODE ANZEIGEN

```
Nicht sarkastischer Artikel:  0
senate panel unanimously approves chris wray's nomination as fbi director
WASHINGTON, July 20 (Reuters) - The U.S. Senate  Judiciary Committee on Thurs
```

Ein Blick in die Daten zeigt, dass in den meisten Fällen die sarkastischen Artikel viel kürzer als die nicht sarkastischen sind (siehe auch Beispiel). Das hängt bei den gegebenen Daten mit der Datenquelle zusammen, d.h. das Portal *theOnion.com* schreibt generell kürzere Artikel, die sarkastisch sind, während auf anderen Seiten sarkastische Texte auch länger sein können. Das Portal *huffingtonpost.com* schreibt hingegen längere Artikel, während andere seriöse Nachrichtenseiten kürzere Artikel schreiben könnten. Das führt dazu, dass dieses Feature nicht genutzt werden kann, da es unbemerkt zu einem Overfitting kommen würde, wenn sich der Klassifikator auf dieses Feature bezieht.

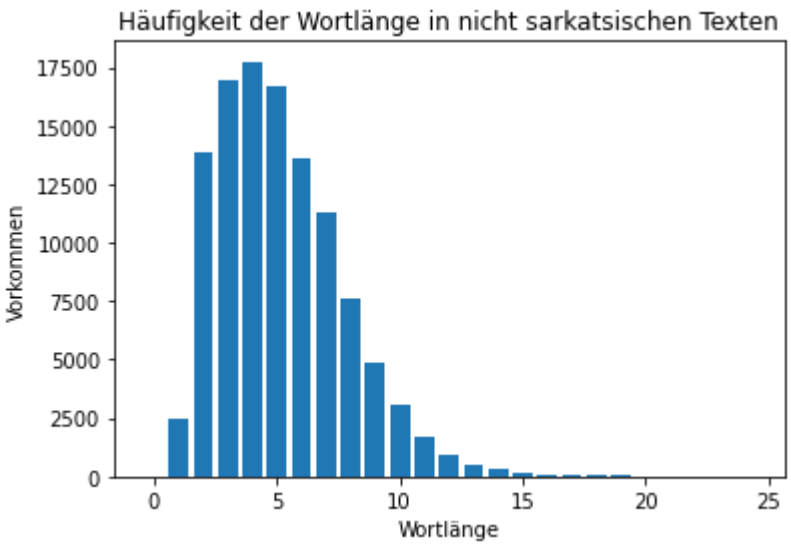
Die Überschriften, sowohl zu sarkastischen als auch nicht sarkastischen Texten, in diesem Datensatz sind etwa gleich lang. Da dieses Projekt darauf abzielt, anhand von Überschriften Sarkasmus zu erkennen und der Artikelinhalt nur als Einordnung und Kontext analysiert wird, wird die Textlänge als Feature nicht weiter berücksichtigt, da eine Korrelation zwischen Sarkasmus und der Anzahl der Worte in einer Überschrift als unwahrscheinlich eingestuft wird.

Wortlänge

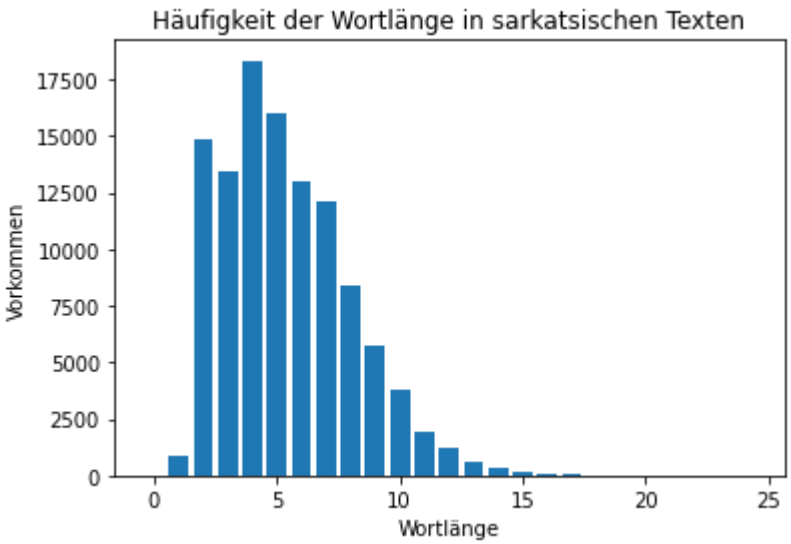
CODE ANZEIGEN

```
Durchschnittliche Wortlänge bei nicht sarkastischen Überschriften:  5.27
Durchschnittliche Wortlänge bei sarkastischen Überschriften:  5.5
```

CODE ANZEIGEN



CODE ANZEIGEN



Die Analyse der Wortlänge führte ebenfalls dazu, dass dieses Feature verworfen werden musste, da kein signifikanter Unterschied festgestellt werden konnte.

Sentimentanalyse

In der Sentimentanalyse werden die Texte auf Polarität (polarity) geprüft, wobei diese einem Wert zwischen -1.0 und +1.0 zugeordnet wird. -1.0 steht dabei für ein sehr negatives Statement und +1.0 für ein sehr positives. Außerdem wird die Subjektivität (subjectivity) auf einer Skala von 0.0 bis 1.0 bestimmt. 1.0 ist dabei sehr subjektiv und 0.0 sehr objektiv. Die Bestimmung des Sentiments könnte bei der Erkennung von Sarkasmus helfen. Dies wäre z.B. dann der Fall, wenn

sarkastische Texte häufiger subjektiv und/oder polarisiert geschrieben wurden oder wenn die Subjektivität oder Polarität dort stärker ausgeprägt ist.

CODE ANZEIGEN

Hier wird das Sentiment der Überschrift und des Inhaltes der Artikel bestimmt und als Feature gespeichert. Zusätzlich wird der Zusammenhang mit dem Sarkasmus-Label analysiert. Überschrift und Inhalt werden zusammen betrachtet, da eine erste Untersuchung gezeigt hat, dass die Überschrift alleine oft als neutral aus der Analyse hervor geht und mit dem Inhalt zusammen einen möglicherweise aussagekräftigeren Wert produziert.

CODE ANZEIGEN

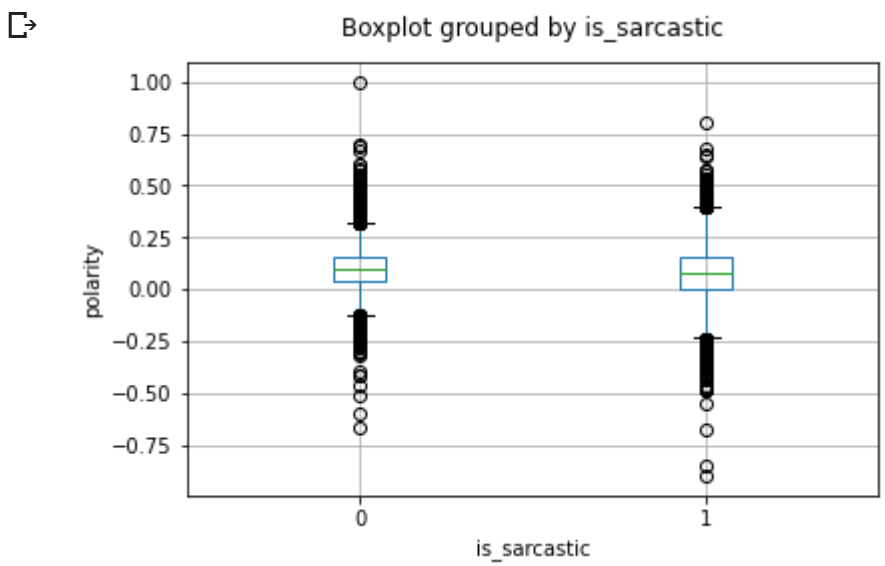
Im Folgendem werden die Ergebnisse als Box-Plot-Diagramme dargestellt, um mögliche Zusammenhänge zu verdeutlichen. Es werden Artikelinhalt und -überschrift zusammen verwendet, da die Wortlänge im Artikel selbst hier nicht als Teil des Kontextes betrachtet wird, der aus dem Artikelinhalt extrahiert werden soll.

Der Boxplot zeigt den Median in der Mitte der *Box*, das 25er- und das 75er-Perzentil als untere und obere Seite der *Box* sowieo das Minimum und Maximum. Die schwarzen Kreise sind Ausreißer. Da der Datensatz relativ viele Daten beinhaltet, gibt es auch entsprechend viele Ausreißer.

CODE ANZEIGEN

Boxplot für die Polarität abhängig vom Sarkasmus:

CODE ANZEIGEN

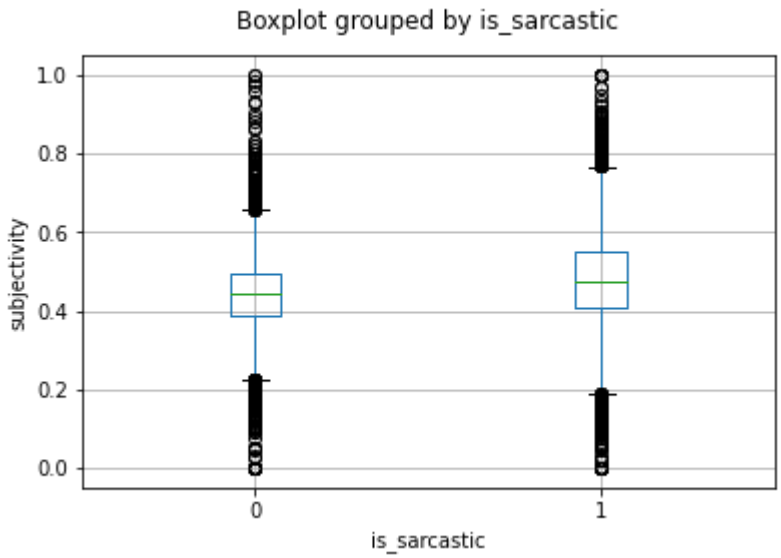


Beim Boxplot der Polarität sieht man zwar, dass das 25er- und 75er-Perzentil sowie die beiden Maxima bei sarkastischen Texten weiter auseinanderliegen, jedoch ist der Unterschied eher marginal. Beim Wert des Medians gibt es noch weniger Abweichung. Das bedeutet, dass die Texte im Durchschnitt wenig polarisierend sind, bei den nicht sarkastischen ist die Polarisierung jedoch noch etwas weniger ausgeprägt. Das zeigt sich auch an der Ausprägung der Ausreißer. Im Bezug auf die vielen Ausreißer stufen wir die Signifikanz der Maxima als eher gering ein, da der größte Teil der Ausreißer und Maxima relativ nahe beieinander liegen.

Aus diesem Boxplot ziehen wir den Schluss, dass die Polarität sich im Allgemeinen zwischen sarkastischen und nicht sarkastischen wenig unterscheidet.

Boxplot für die Subjektivität abhängig vom Sarkasmus:

CODE ANZEIGEN



Der Boxplot der Subjektivität weist hingegen etwas deutlichere Unterschiede auf. Hier unterscheidet sich sowohl der Median, als auch die Distanz zwischen 25er- und 75-er-Perzentil als auch die Distanz zwischen Maximum und Minimun zwischen den Texten. Es gibt also generell eine höhere Subjektivität bei sarkastischen Texten als bei nicht sarkastischen, da hier der Median mehr in Richtung Subjektivität, also dem Wert 1.0, geht. Auch das Maximum bei sarkastischen Texten ist höher als bei nicht sarkastischen. Entgegen der Erwartung ist das Minimum allerdings sogar geringer als bei nicht sarkastischen Texten.

CODE ANZEIGEN

Auch bei der Korrelation zwischen Polarität und Subjektivität bei sarkastischen und nicht sarkastischen Texten lässt sich kein Unterschied feststellen. Die orangen und blauen Punkte sind annähernd deckungsgleich.

Obwohl der Boxplot nur bei der Subjektivität etwas größere Unterschiede aufweist, nehmen wir beide Kriterien mit in das Neuronale Netz auf, falls es einen Zusammenhang gibt, der sich zwar alleine nicht erkennen lässt, aber mit anderen Features zu einer besseren Klassifizierung führt.

Emotionenanalyse

In der Emotionenanalyse werden Emotionen mit Hilfe einer Emotionsliste gefunden, die bestimmten Schlagwörtern eine Emotion zuschreibt. Wenn im eingegebenen Text diese Schlagwörter vorkommen, dann kann die dahinter stehende Emotion dem Text zugeordnet werden.

Auch hier liegt die Annahme zu Grunde, dass sarkastische Texte stärker emotionalisiert sind als nicht sarkastische. Sollte sich dies bewahrheiten, kann die Analyse die Klassifizierung verbessern.

Die Gefahr bei diesem Verfahren ist, dass die Emotionsliste zu wenig Schlagwörter (und Varianten dieser Wörter abdeckt) und nicht genügend Emotionen erkannt und zugeordnet werden können. Dadurch kann die Analyse an Aussagekraft verlieren

CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

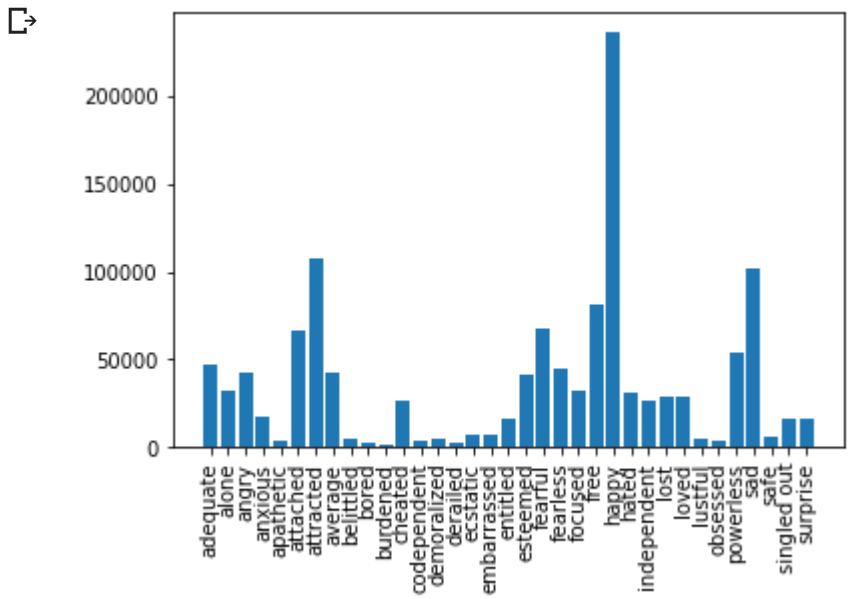
CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

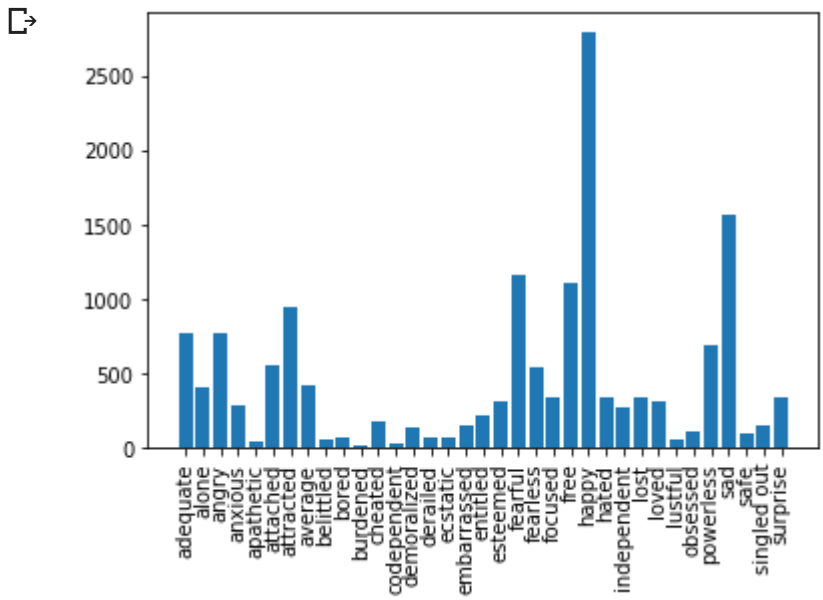
Emotionen in nicht sarkastischen Artikeln:

CODE ANZEIGEN



Emotionen in sarkastischen Artikeln:

CODE ANZEIGEN



Wie werden Ergebnisse der Emotionsanalyse verwendet?

Anhand der Grafiken lässt sich erkennen, dass die Emotionen sowohl in sarkastischen als auch nicht sarkastischen Texten ähnlich verteilt sind. Allerdings unterscheidet sich die Anzahl der Emotionen stark. Das liegt vor allem daran, dass sarkastische Artikel eher kürzer sind als nicht sarkastische. Die Länge des Textes soll aber nicht zur Klassifikation von Sarkasmus verwendet werden, da der Fokus dieses Projektes darauf liegt, Sarkasmus anhand einer Überschrift zu

erkennen. Die Annahme, dass sarkastische Texte mehr Emotionen erhalten, kann der gegebene Datensatz nicht bestätigen.

Sowohl in nicht sarkastischen als auch sarkastischen Artikeln und Überschriften finden sich zudem allgemein sehr wenig Emotionsschlagworte. Daher werden die Emotionen aus Überschrift und Artikel zusammen betrachtet.

Aufgrund der wenigen Vorkommnisse von Emotionen, haben wir uns dazu entschieden, auch die Anzahl der Emotionen nicht zu berüchsichtigen, sondern als Feature nur aufzunehmen, welche Emotionen einem Artikel zugeordnet sind.

Slanganalyse

In sarkastischen Texten wird möglicherweise häufiger Umgangssprache/Slang verwendet, als in nicht sarkastischen Texten. Allerdings liegt auch die Vermutung nahe, dass in den Artikelüberschriften aufgrund der Kürze wenig bis gar kein Slang zu finden ist.

Da es keine frei verfügbare Slang-Library gibt, die herausfinden kann, ob ein Text in Slang geschrieben wurde oder nicht, muss dies anhand bestimmter, selbst gewählter Eigenschaften herausgefunden werden.

Im Folgenden wird untersucht, ob ein Text profan ist oder nicht. Dafür wurde die Library *profanity-check* verwendet, damit dennoch herausgefunden werden kann, ob ein Text anstößige Worte enthält oder nicht.

Um eine verbesserte Slanganalyse zu gewährleisten, könnten nun noch mehr Schlagworte ermittelt werden, in der Art, wie die Emotionen bereits ermittelt wurden. Es gibt also eine Liste mit Triggerworten, die angeben, ob es sich um ein Slang-Wort handelt oder nicht. Das Problem bei der Slangerkennung selbst ist, dass z.B. der Dialekt oder die Altersgruppe eine Rolle spielen kann und wir uns deswegen von Schlagworten keine umfassend gute Klassifizierung versprechen. Es könnte zum Beispiel sein, dass es ein gutes Dictionary für den einen Dialekt gibt, für einen anderen Dialekt ist das Wörterbuch aber nicht so umfassend. Damit würde Sarkasmus in einem Dialekt besser klassifiziert werden als in einem anderen. Zudem gibt es immer wieder Wortneuschöpfungen und eine Veränderte Bedeutung eines bestehenden Wortes, welches das Ergebnis und somit die Klassifikation verfälschen kann.

Die verwendete Library *profanity-check* nutzt kein Dictionary zum Finden bestimmter Wörter, sondern eine SVM und verspricht damit eine umfassendere und performantere Analyse profaner Wörter. Als Output liefert sie einen Wert zwischen 0.0 und 1.0 zurück, welcher beschreibt, wie beleidigend ein Text ist. 0.0 ist dabei nicht profan und 1.0 ist sehr profan.

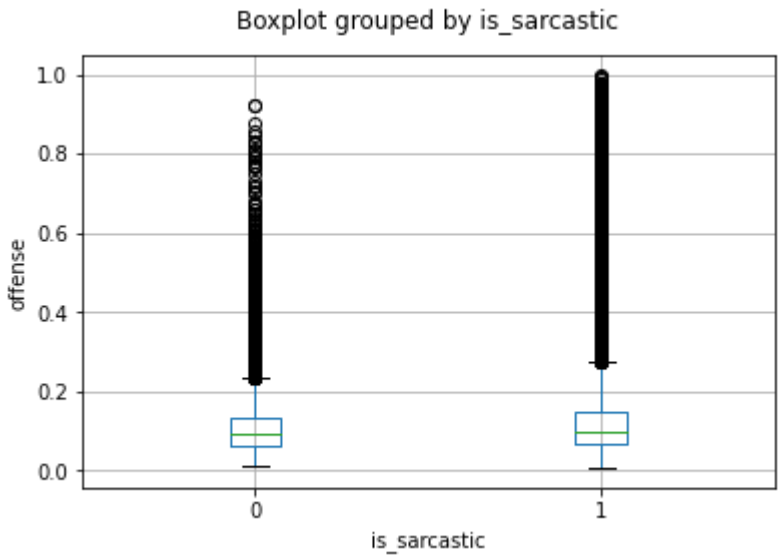
CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

Anstößigkeit Überschrift

CODE ANZEIGEN



In den folgenden Diagrammen wird die Anstößigkeit in den Überschriften und den jeweiligen Inhalten der Artikel betrachtet.

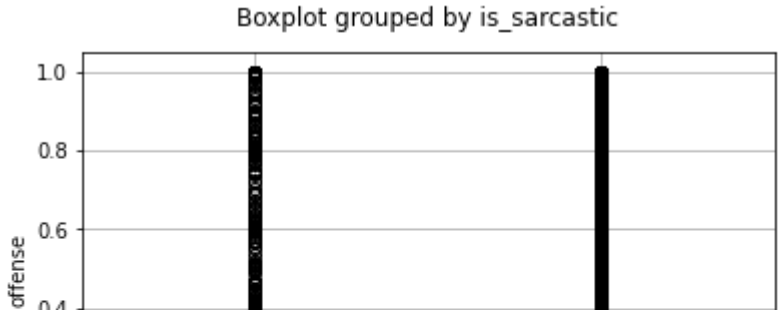
Bei der Anstößigkeit in den Überschriften ist erkennbar, dass sarkastische Texte eher profan sind als nicht sarkastische, obgleich beide Texte im Allgemeinen wenig profan sind.

Das Minimun liegt bei beiden Diagrammen nahe bei 0.0. Auffällig sind jedoch die Ausreißer, die zum einen bei beiden Diagrammen nur überhalb des Maximums vorkommen, bei sarkastischen Texten haben Überschriften aber auch Werte 1.0, während dies bei nicht sarkastischen Texten nicht vorkommt.

Daraus könnte man folgenden Schluss ziehen: ist ein Text sehr profan geschrieben, ist er mit höherer Wahrscheinlichkeit sarkastisch. Ist er das nicht, liefert das aber keine zuverlässige Aussage.

Anstößigkeit Inhalt

CODE ANZEIGEN



Die Anstößigkeit in den Artikelinhalten fällt bei nicht sarkastischen Artikeln extrem niedrig aus. Der Median, das 25-er und 75-Perzentil sowie Minimum und Maximun liegen so nahe bei 0.0 und so nahe beieinander, dass man sie nicht sehen kann. Ausreißer finden sich ebenfalls nahe bei 0, allerdings ist das Maximum der Ausreißer auch bei 1.0 angesiedelt. Man kann aber im Vergleich zu den Ausreißern bei sarkastischen Texten anhand der Färbung sehen, dass es bei nicht sarkastischen Texten weniger Ausreißer gibt. Obwohl der Median bei nicht sarkastischen Texten immer noch nahe 0.0 liegt, unterscheidet sich das Maximum hier vergleichsweise stark von dem Maximum der nicht sarkastischen Texte.

Dies scheint also ein guter Indikator für sarkastische Texte zu sein. Andersrum gilt dies aber nicht zwangsläufig. D.h. beinhaltet ein Text anstößige Inhalte, ist die Wahrscheinlichkeit höher, dass es ein sarkastischer Text ist, beinhaltet er aber keine profanen Begriffe, ist das kein Indikator für einen nicht sarkastischen Text, da die generelle Anstößigkeit trotz allem sehr niedrig ausfällt.

Methoden zur Klassifizierung von Sarkasmus

Für Klassifizierungsprobleme, wie das Erkennen von Sarkasmus, gibt es im Gebiet des Maschinellen Lernens verschiedene Verfahren. In diesem Projekt beschränken wir uns auf ein Neuronales Netz und zum Vergleich eine Support Vector Machine. Als Machine Learning Framework wurden *Keras* und *scikit-learn* ausgesucht.

Neuronales Netz mit Keras

Das folgende Vorgehen ist angelehnt an [Practical Text Classification With Python and Keras].

Preprocessing

Worte, wie sie in den Überschriften vorkommen, können nicht direkt in das Neuronale Netz gegeben werden, da den einzelnen Buchstaben in einem Wort keine Bedeutung entnommen werden kann. Die meisten Algorithmen erwarten zudem einen numerischen Vektor mit einer festen Größe. Daher müssen die Worte in Zahlen umgewandelt werden, damit das neuronale Netz sie verarbeiten kann.

Mit *scikit-learn* können mit dem Count Vectorizer Texte transformiert und so in ein neuronales Netz gegeben werden. Zunächst wird den einzelnen Worten ein numerischer Token zugeordnet, anschließend werden die Tokens im Text gezählt und anhand der Anzahl der Vorkomnisse gewichtet. Dieses Verfahren nennt man auch Bag-of-Words. Es wird häufig in Analysen von Texten angewandt.

Das Model

Beim Bau unseres Models haben wir uns an das allgemeine Vorgehen gehalten, für die Hidden Layers eine ReLu-Funktion (rectified linear unit) als Aktivierungsfunktion zu nutzen und eine Sigmoidfunktion für den Output Layer, da es sich bei unserem Problem um eine binäre Klassifikation (sarkastisch oder nicht sarkastisch) handelt. Außerdem haben wir die Batchgröße auf 10 festgelegt, da dies ein allgemeiner Standard für die Anfangsgröße ist und auch bei Veränderungen keine Verbesserung im Test erzielt wurde.

CODE ANZEIGEN

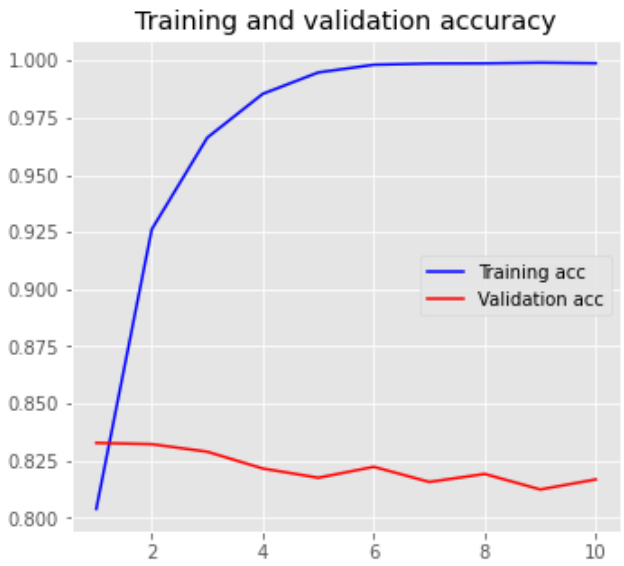
CODE ANZEIGEN

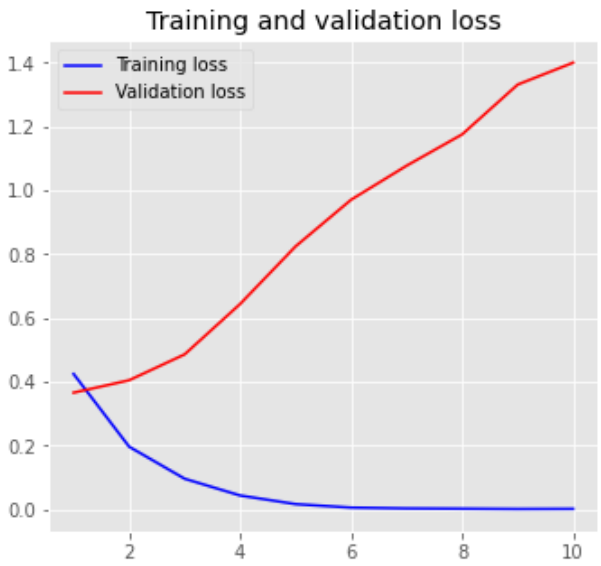
CODE ANZEIGEN

Erster Test

CODE ANZEIGEN

```
Training Accuracy: 0.9993
Testing Accuracy:  0.8169
```





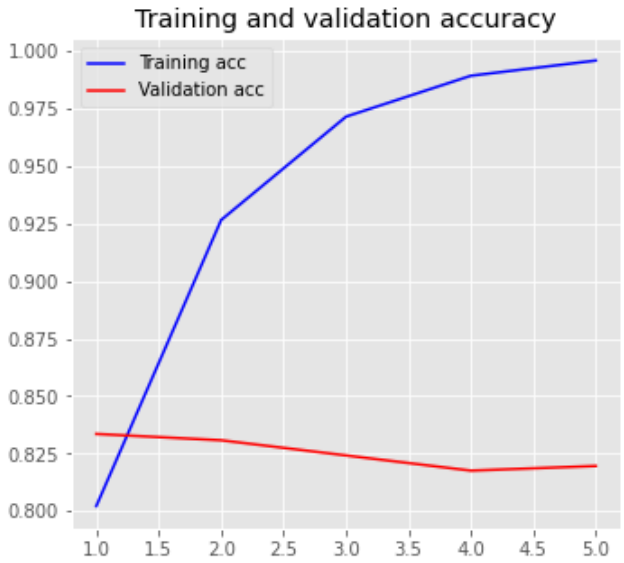
Hidden Layer und Epochen

Auf der Suche nach dem besten Model wird im Folgenden zum Vergleich die Anzahl der Hidden Layer und die Anzahl der Epochen variiert.

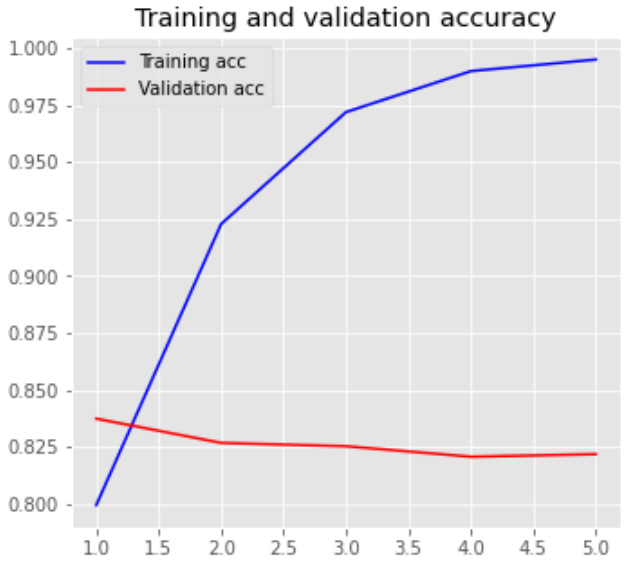
CODE ANZEIGEN

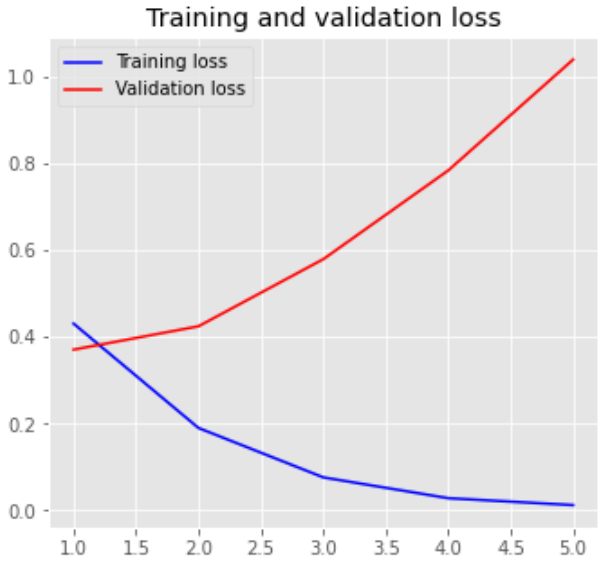
CODE ANZEIGEN

```
Number hidden layer: 3    Number epochs: 5
Training Accuracy: 0.9980
Testing Accuracy:  0.8196
```

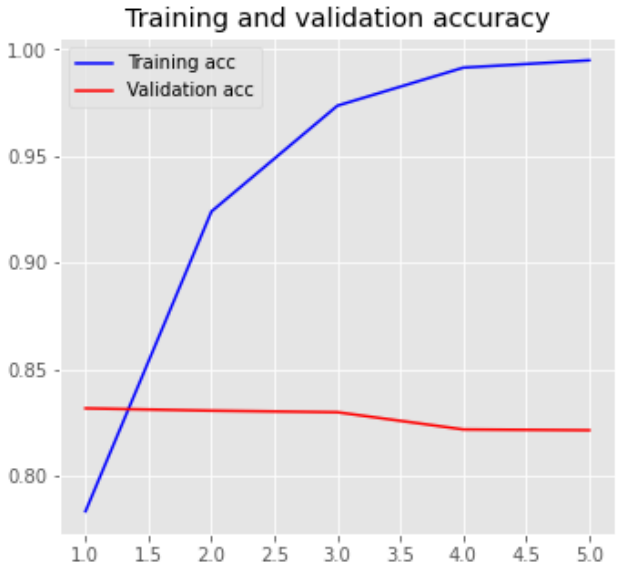


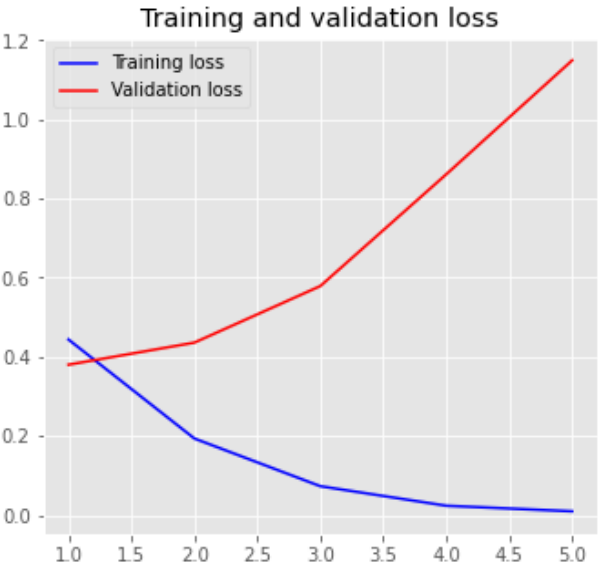

```
Number hidden layer: 6    Number epochs: 5
Training Accuracy: 0.9976
Testing Accuracy:  0.8218
```





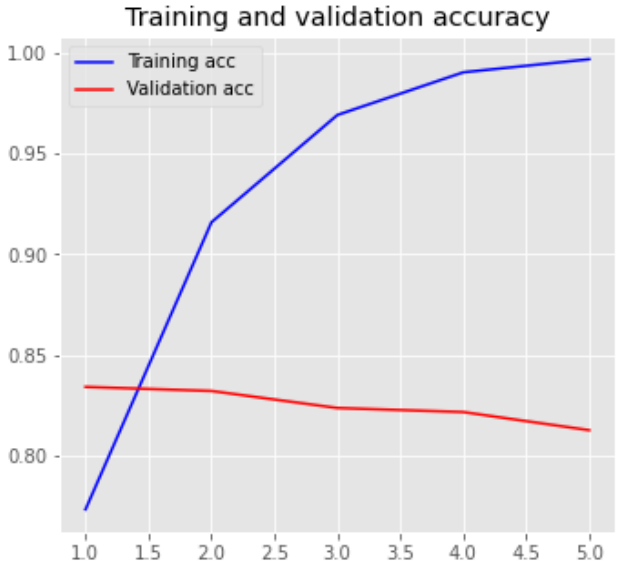

```
Number hidden layer: 9    Number epochs: 5
Training Accuracy: 0.9971
Testing Accuracy:  0.8213
```

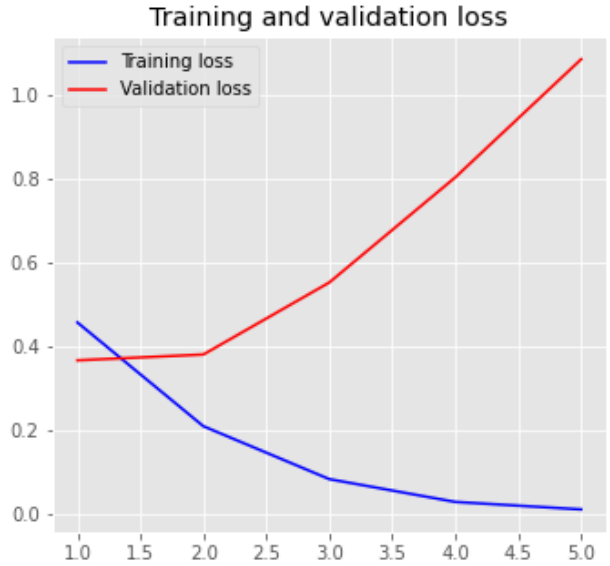




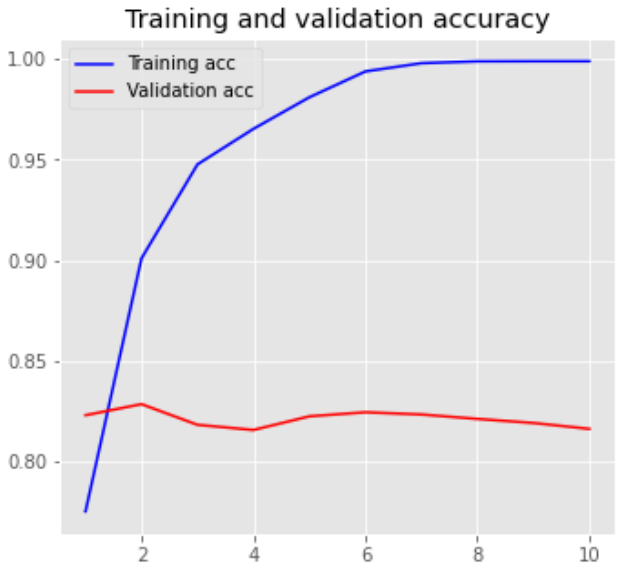

```
Number hidden layer: 12    Number epochs: 5
```

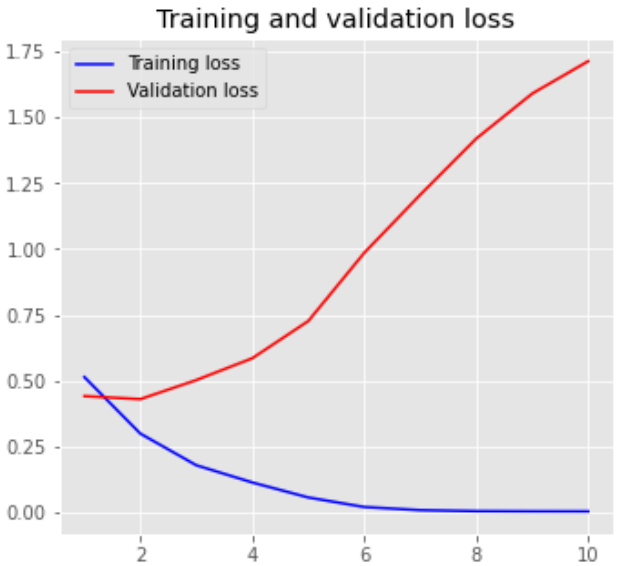
```
Training Accuracy: 0.9982
Testing Accuracy:  0.8127
```



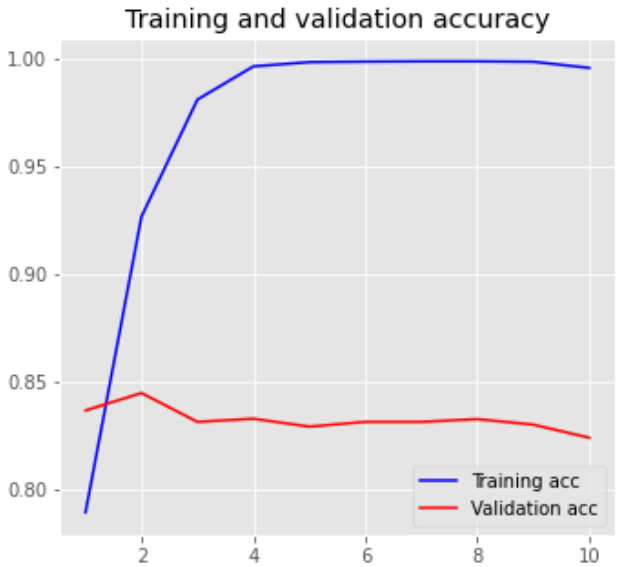


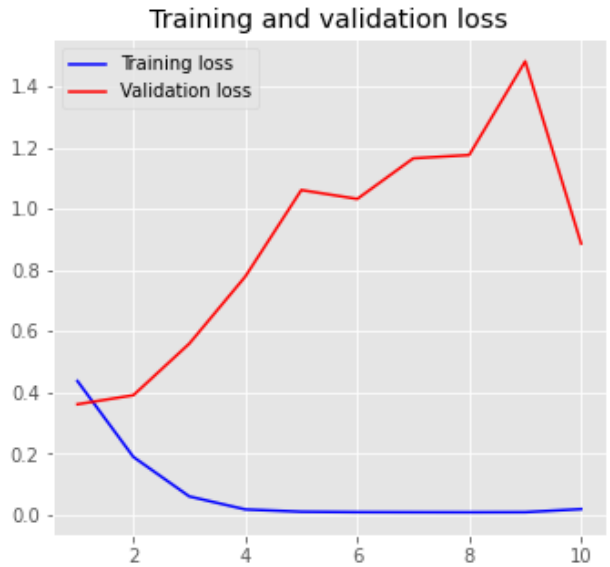

```
Number hidden layer: 3    Number epochs: 10
Training Accuracy: 0.9991
Testing Accuracy:  0.8161
```





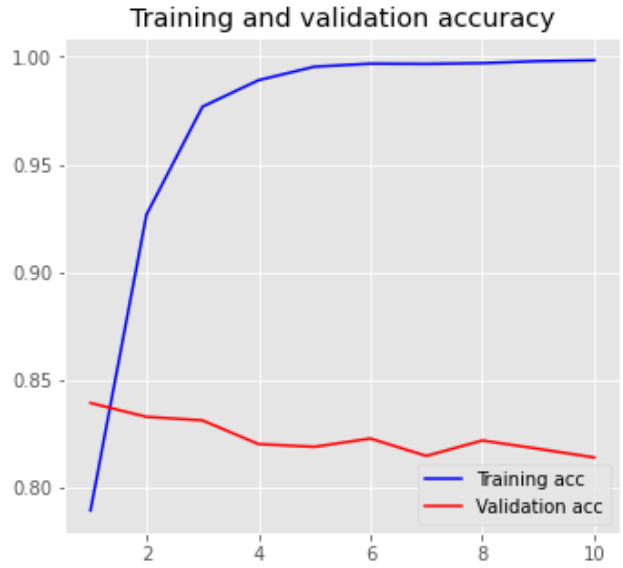

```
Number hidden layer: 6    Number epochs: 10
Training Accuracy: 0.9913
Testing Accuracy:  0.8240
```

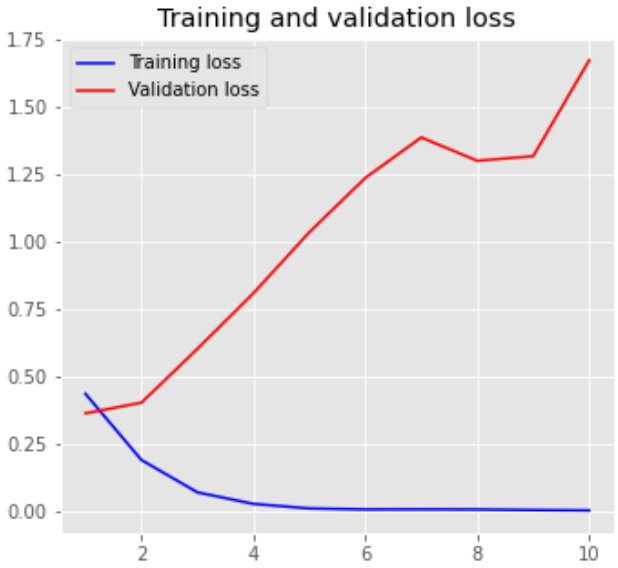




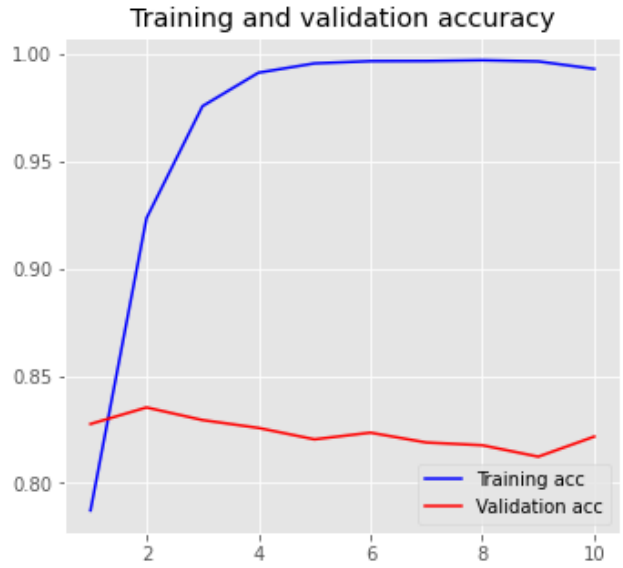

```
Number hidden layer: 9    Number epochs: 10
Training Accuracy: 0.9982
Testing Accuracy:  0.8138
```

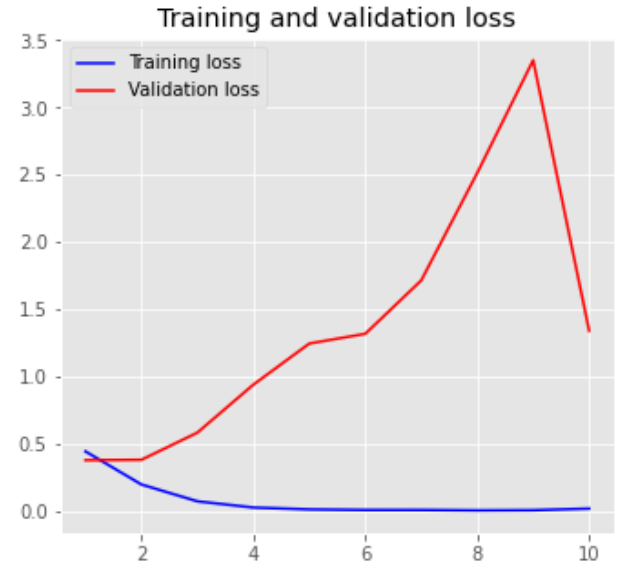
Testing Accuracy: 0.8138



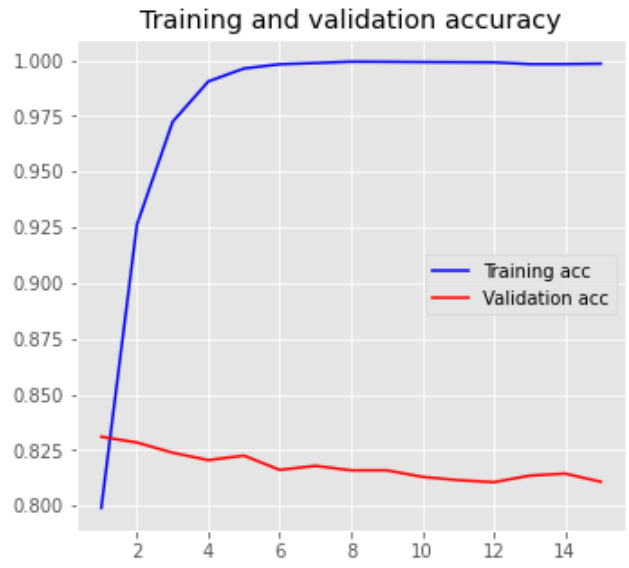


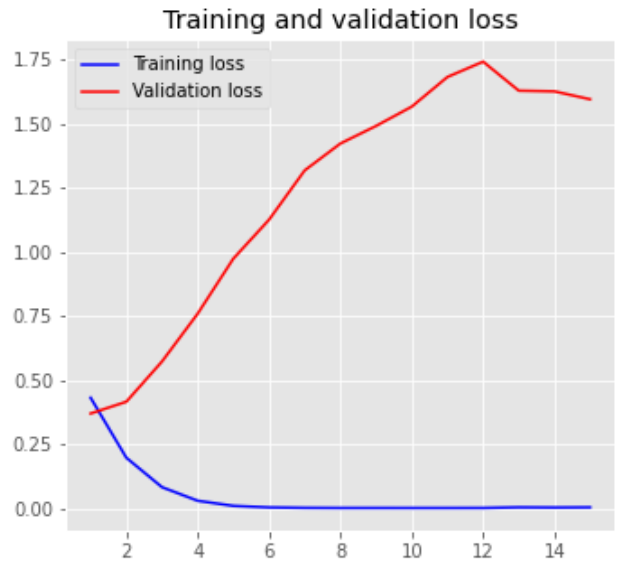
Number hidden layer: 12 Number epochs: 10
Training Accuracy: 0.9954
Testing Accuracy: 0.8217



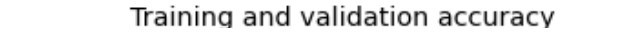


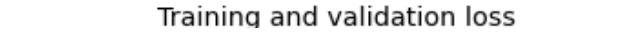
Number hidden layer: 3 Number epochs: 15
Training Accuracy: 0.9991
Testing Accuracy: 0.8108

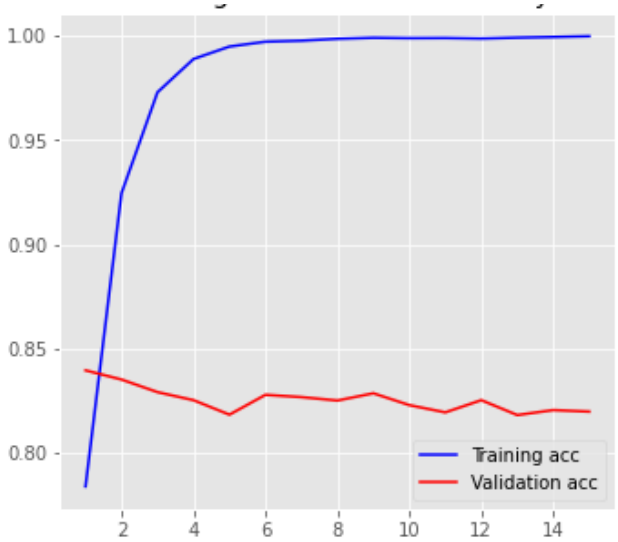




Number hidden layer: 6 Number epochs: 15
Training Accuracy: 0.9999
Testing Accuracy: 0.8196

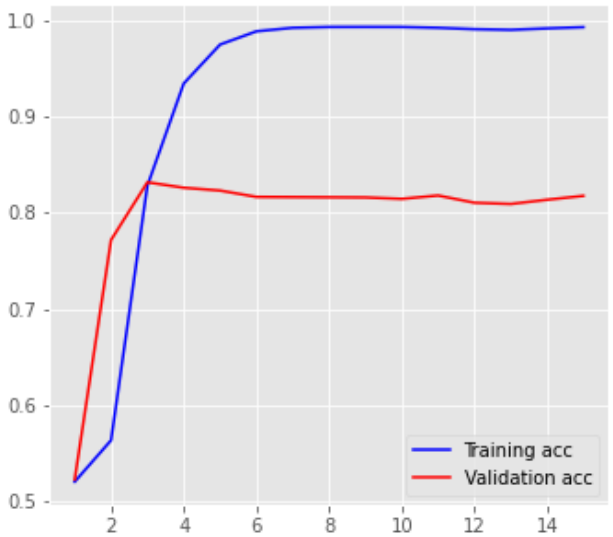







```
Number hidden layer: 9    Number epochs: 15
Training Accuracy: 0.9966
Testing Accuracy:  0.8174
```

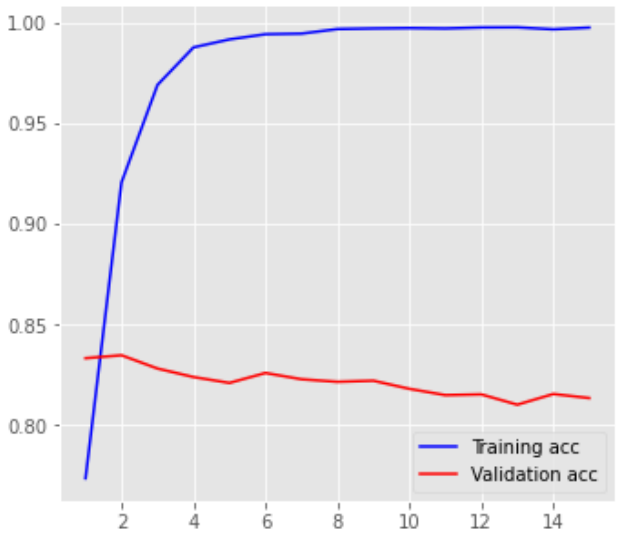
Training and validation accuracy



Training and validation loss


```
Number hidden layer: 12    Number epochs: 15
Training Accuracy: 0.9976
Testing Accuracy:  0.8136
```

Training and validation accuracy

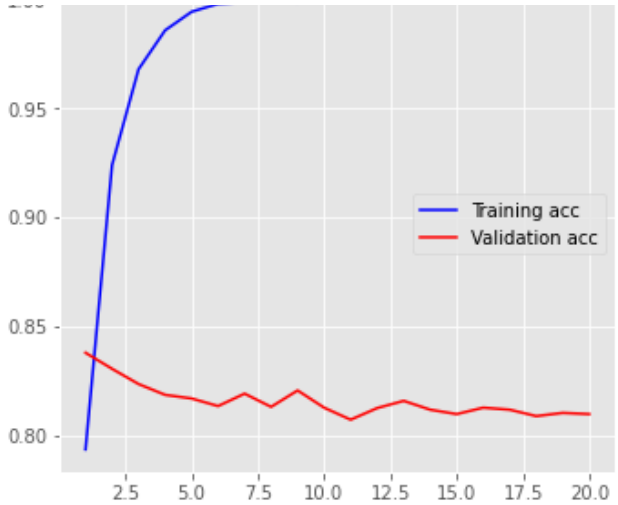


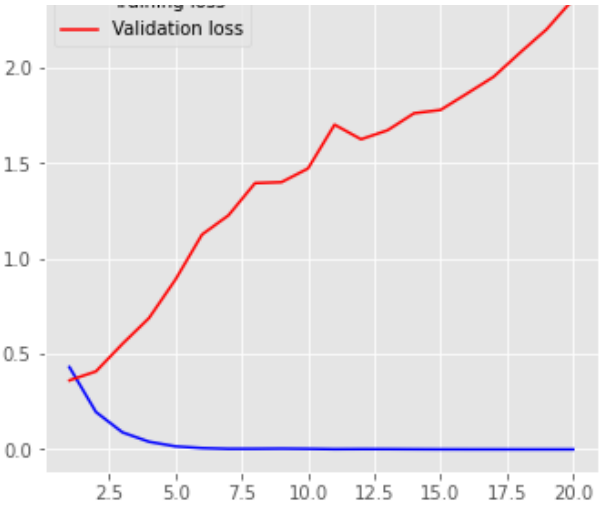
Training and validation loss


```
Number hidden layer: 3    Number epochs: 20
Training Accuracy: 1.0000
Testing Accuracy:  0.8101
```

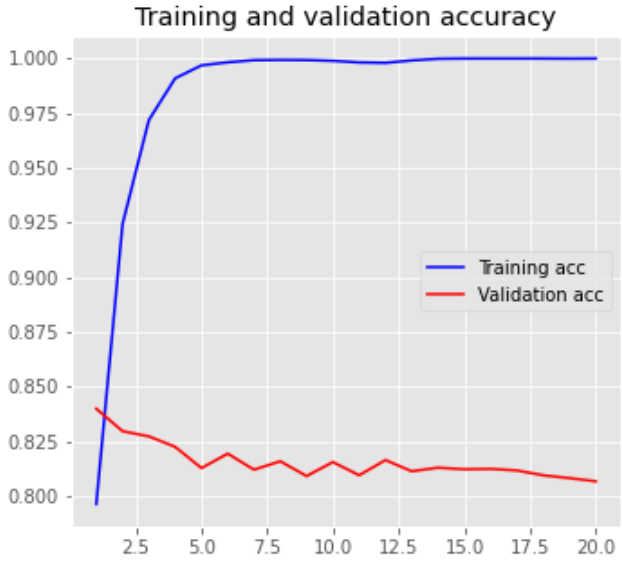
Training and validation accuracy

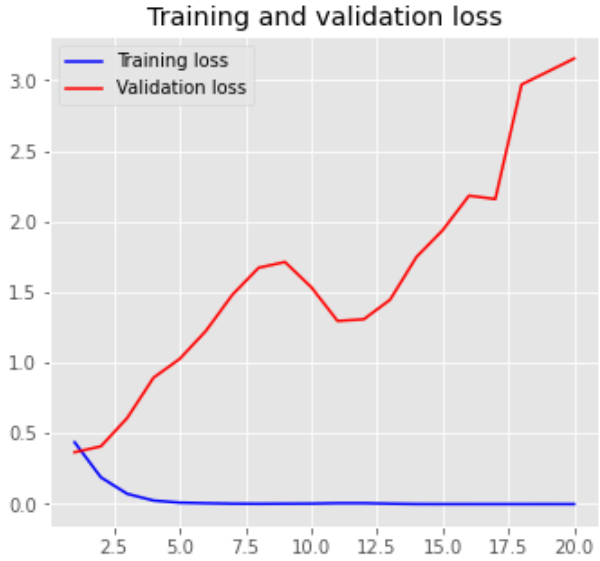
Training and validation loss



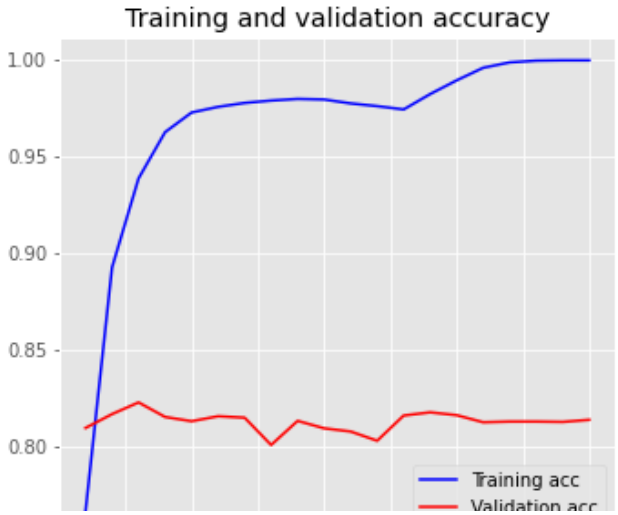


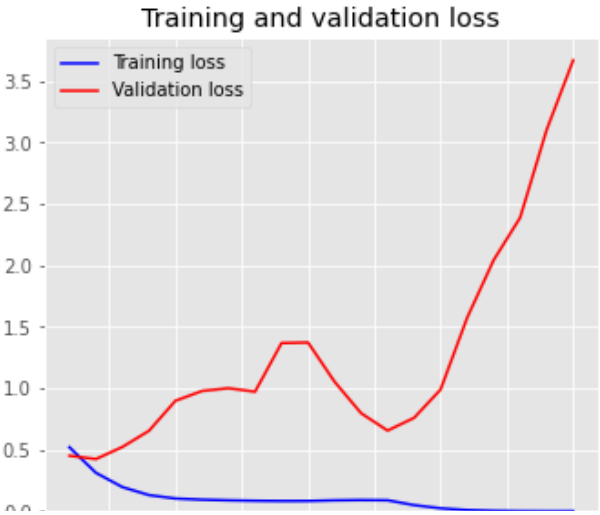

```
Number hidden layer: 6    Number epochs: 20
Training Accuracy: 1.0000
Testing Accuracy:  0.8066
```






```
Number hidden layer: 9    Number epochs: 20
Training Accuracy: 0.9994
Testing Accuracy:  0.8141
```





Beim Vergleich der verschiedenen Modelle fällt auf, dass zu viele Hidden Layers eine Verschlechterung darstellen und zu viele Epochen zu Overfitting führen.

Außerdem ist auffällig, dass mit 12 hidden Layers und 20 Epochen der Genauigkeitswert von über 81% auf etwas knapp über 52% einbricht, und das sowohl bei den Trainings- als auch

Testdaten. Dieses Verhalten haben wir nicht erwartet und können uns auch nicht erklären, insbesondere bei den Trainingsdaten.

Folgende Parametrisierungen wollen wir zum Vergleich heranziehen. Damit kann damit die Anzahl der Epochen sowie die Anzahl der Hidden Layer und deren Einfluss auf die Genauigkeit bei Test und Training festgestellt werden.

- Number hidden layer: 12 Number epochs: 20 Training Accuracy: 0.5210 Testing Accuracy: 0.5226
- Number hidden layer: 9 Number epochs: 20 Training Accuracy: 0.9994 Testing Accuracy: 0.8141
- Number hidden layer: 12 Number epochs: 15 Training Accuracy: 0.9976 Testing Accuracy: 0.8136

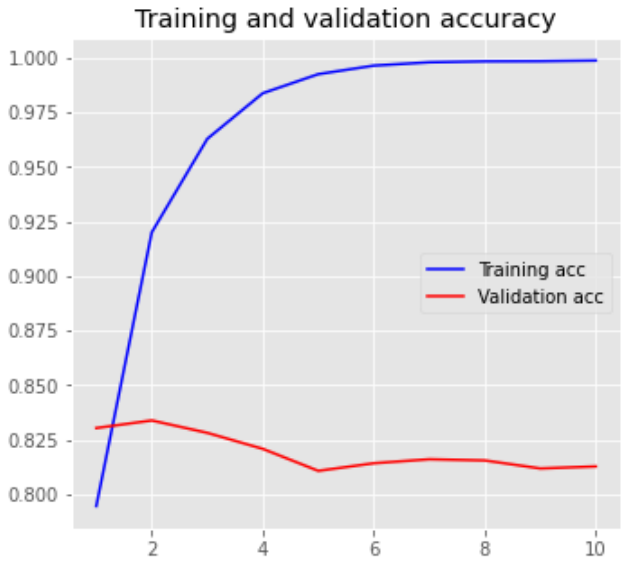
Diese Werte sehen sehr ähnlich aus, und vor allem zu allen anderen Werten, die produziert wurden. Es gibt keine wesentliche Tendenz zu einem schlechtenren Wert. Mit mehr Epochen würden wir zudem eher ein Overfitting erwarten, also eine Annäherung der Trainings-Genauigkeit gegen 100%, statt den beobachteten Einbruch.

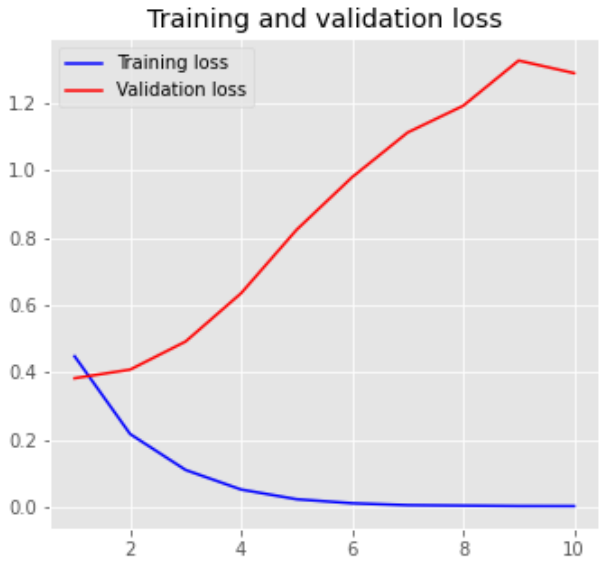
Das beste getestete Model hat 10 Epochen und 6 Hidden Layer. Dieses Model wird für die nächste Einstellungen verwendet.

Neuronenanzahl

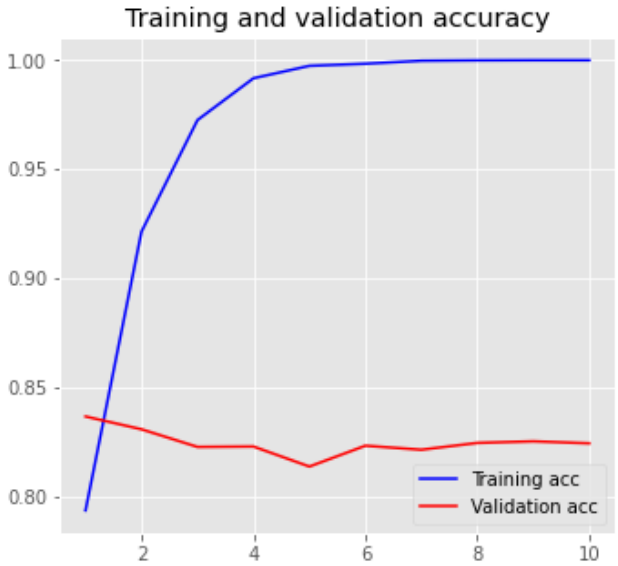
CODE ANZEIGEN

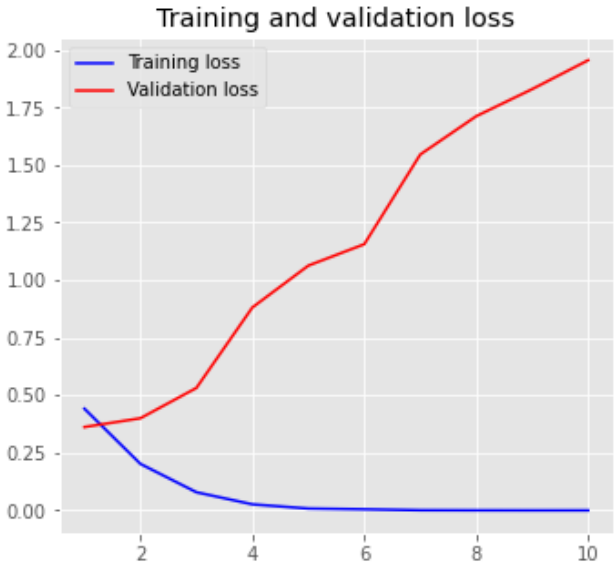
```
Number neurons: 5
Training Accuracy: 0.9990
Testing Accuracy:  0.8127
```



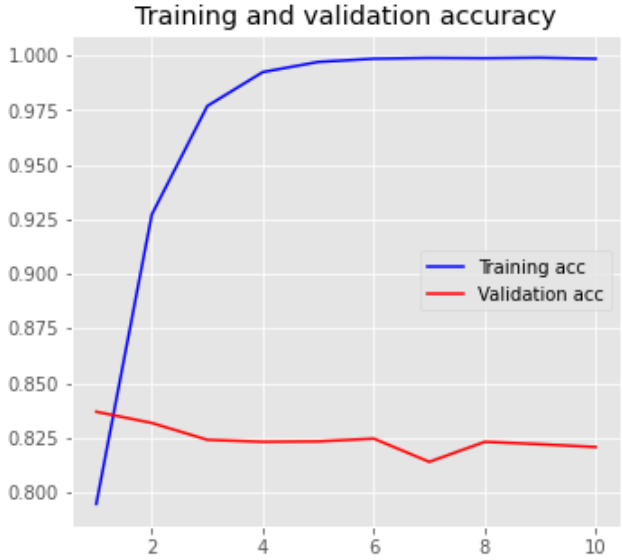


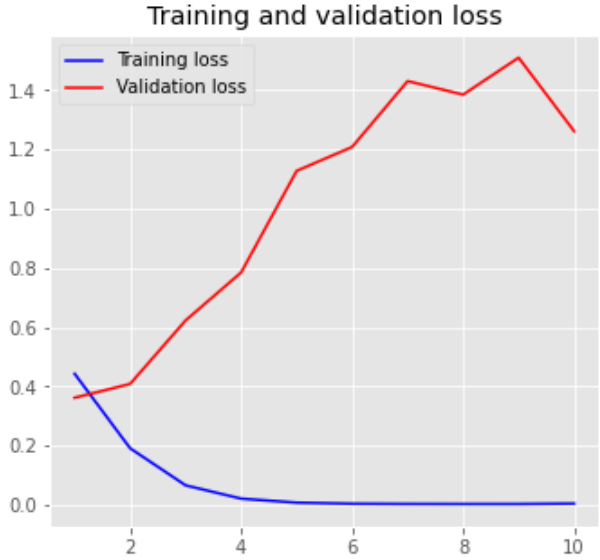

```
Number neurons: 8
Training Accuracy: 1.0000
Testing Accuracy:  0.8246
```





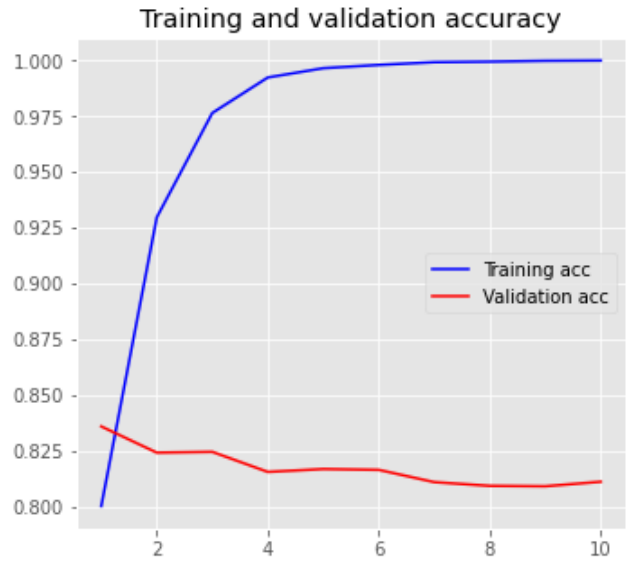

```
Number neurons: 11
Training Accuracy: 0.9988
Testing Accuracy:  0.8207
```



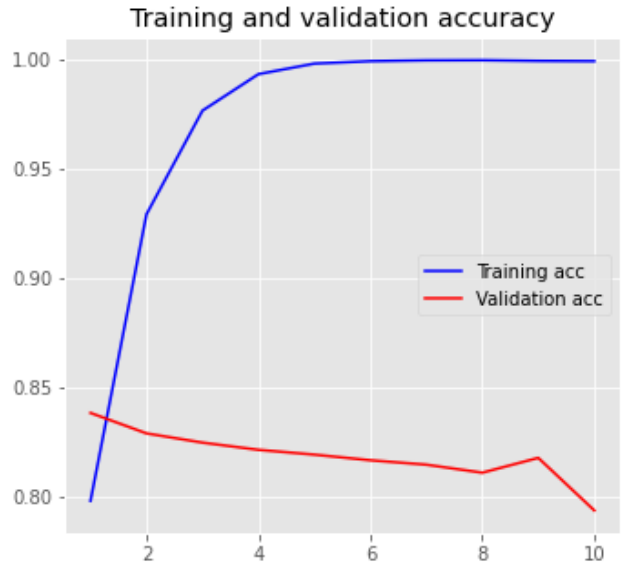


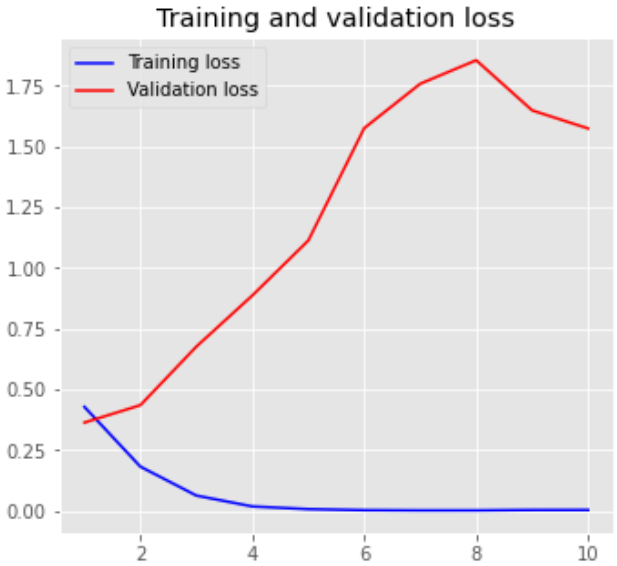

```
Number neurons: 14
```

```
Training Accuracy: 1.0000
Testing Accuracy:  0.8114
```

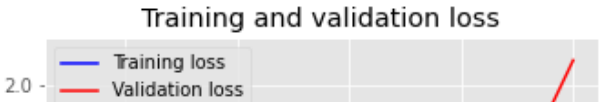



```
Number neurons: 17
Training Accuracy: 0.9867
Testing Accuracy:  0.7936
```






```
Number neurons: 20
Training Accuracy: 1.0000
Testing Accuracy:  0.8233
```



Die Veränderung bei der Anzahl der Neuronen in einem Layer zeigt, dass die Einstellung von 10 Neuronen im optimalen Bereich liegt. Sowohl weniger, als auch mehr Neuronen liefern kein wesentlich besseres Ergebnis.

Optimierungsfunktion

Keras bietet verschiedene Optimierungsfunktionen. Diese Funktionen beeinflussen das Lernverhalten des Neuronalen Netzes und soll das Gradientenverfahren verbessern. Folgende Funktionen wurden zum Test ausgewählt:

- Adam

- Nadam
- Adamax

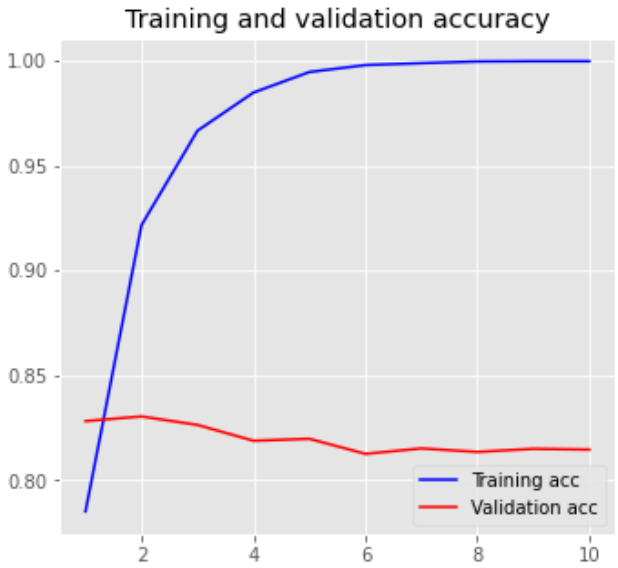
Adam ist eines der meistgenutzten Verfahren. *Nadam* und *Adamax* erweitern dieses auf unterschiedliche Weisen.

Im Folgenden werden die Optimierungsfunktionen am ersten Testfall getestet.

CODE ANZEIGEN

CODE ANZEIGEN

```
Adam:
Training Accuracy: 0.9996
Testing Accuracy:  0.8147
```




```
Nadam:
```

Obwohl das Neuronale Netz nur ein Feature, nämlich die vektorisierten Worte, bekommt, liefert das Neuronale Netz mit zwischen 80% und 83% Genauigkeit bei den Testdaten ein gutes Ergebnis. Dabei wurden an mehreren Parametern Anpassungen gemacht.

- Anzahl der Epochen: anhand der erstellen Diagramme der Trainingshistorie lässt sich erkennen, dass eine geringe Anzahl an Epochen zu einer besseren Genauigkeit bei der Klassifizierung der Testdaten führt. Je höher die Epochenanzahl, desto mehr schlägt der Graph der Loss-Funktion über die Testdaten aus.
- Anzahl der Hidden Layer: Sind zu viele Hidden Layer im Neuronalen Netz, verringert sich die Genauigkeit bei der Klassifizierung der Testdaten wieder ein wenig.
- Optimizer: Es wurden verschiedene Optimierungsfunktionen getestet. Dabei stellte sich heraus, dass Adamax die höchste Genauigkeit zur Klassifizierung der Testdaten liefert.

Neuronales Netz mit mehreren Features

Das folgende Vorgehen ist angelehnt an [Building a mixed-data neural network in Keras to predict accident locations].

Im Folgenden wird ein neuronales Netz mit folgenden Features trainiert:

- Überschrift
- Beleidigungen/Anstößigkeit
- Emotionen
- Sentiments (Polarität und Subjektivität)

Da die Features verschiedene Formate haben werden sie aufgeteilt und als seperate Inputs für drei Sub-Modelle verarbeitet. Das ist zunächst das Model für die Überschriften, welche in Textform vorliegen, das Model für die Emotionen, welche ebenfalls in Textform vorliegen und das Model für die numerischen Werte der Sentiments und der Anstößigkeit. Die drei beschriebenen Sub-Modelle werden zu einem Model verbunden, indem ihre Outputs in weiteren Neuronen verarbeitet werden und zu der finalen Outputschicht und der Klassifizierung führen. Das gesamte Model wird als eins trainiert, sodass ihre Werte gemeinsam angepasst und eingestellt werden.

CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

Beim Training des Models für das Neuronale Netz mit mehreren Features werden die Erfahrungen aus dem einfachen Model zuvor mit einbezogen. Dort wurde herausgefunden, dass der Optimizer **Adamax** das beste Ergebnis erzielt, daher wurde für das Training dieses Models nur dieser Optimizer verwendet. Angepasst wurden nur die Anzahl der Hidden Layers und die Anzahl der Epochen mit Anlehnung an die besten Ergebnisse aus dem einfachen Model.

CODE ANZEIGEN

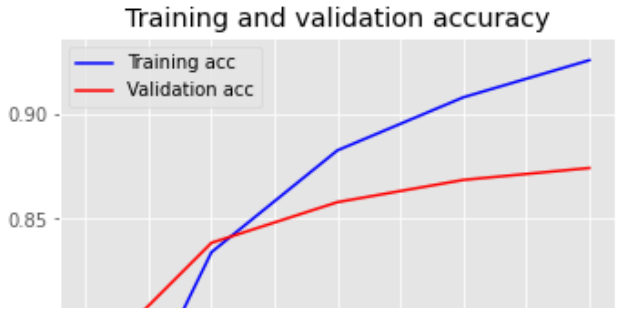
Das Model mit dem Optimizer "Adamax"

CODE ANZEIGEN

5 Epochen, 1 Hidden Layer

CODE ANZEIGEN

```
Epoch 1/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.5593 - acc
Epoch 2/5
1635/1635 [==============================] - 3s 2ms/step - loss: 0.3760 - acc
Epoch 3/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.2892 - acc
Epoch 4/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.2349 - acc
Epoch 5/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.1948 - acc
Training Accuracy: 0.9414
Testing Accuracy:  0.8743
```



10 Epochen, 1 Hidden Layer

CODE ANZEIGEN

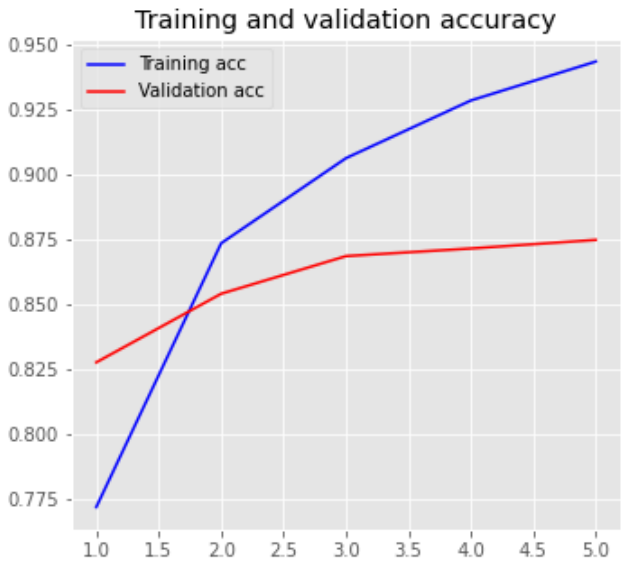
```
Epoch 1/10
1635/1635 [==============================] - 4s 2ms/step - loss: 0.6042 - acc
Epoch 2/10
1635/1635 [==============================] - 4s 2ms/step - loss: 0.4372 - acc
Epoch 3/10
1635/1635 [==============================] - 4s 2ms/step - loss: 0.3462 - acc
Epoch 4/10
1635/1635 [==============================] - 3s 2ms/step - loss: 0.2799 - acc
Epoch 5/10
```

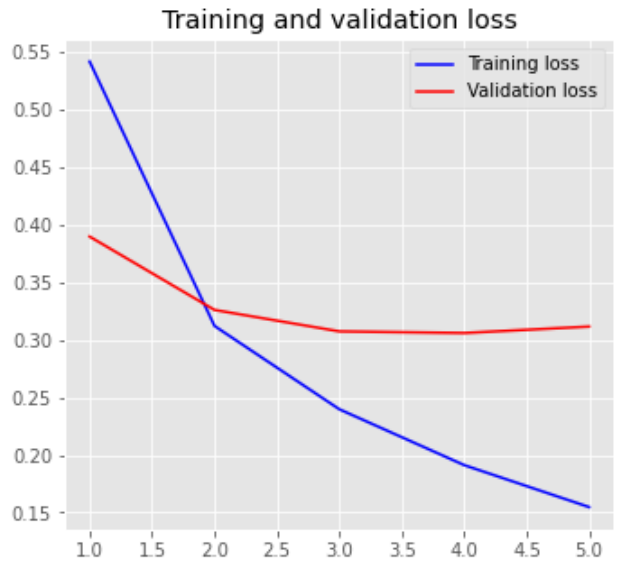
5 Epochen, 2 Hidden Layers

```
1635/1635 [==============================] - 4s 2ms/step - loss: 0.1915 - acc
```

CODE ANZEIGEN

```
Epoch 1/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.5413 - acc
Epoch 2/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.3119 - acc
Epoch 3/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.2396 - acc
Epoch 4/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.1911 - acc
Epoch 5/5
1635/1635 [==============================] - 4s 2ms/step - loss: 0.1547 - acc
Training Accuracy: 0.9563
Testing Accuracy:  0.8745
```

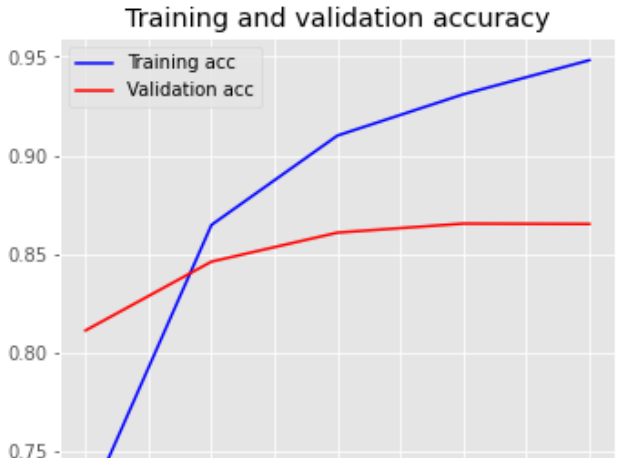


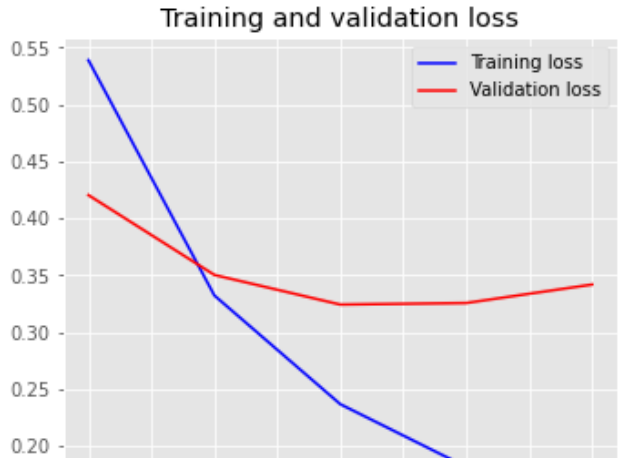


5 Epochen, 2 Hidden Layers and 4 Hidden Layers

CODE ANZEIGEN

```
Epoch 1/5
1635/1635 [==============================] - 4s 3ms/step - loss: 0.5389 - acc
Epoch 2/5
1635/1635 [==============================] - 4s 3ms/step - loss: 0.3324 - acc
Epoch 3/5
1635/1635 [==============================] - 4s 3ms/step - loss: 0.2369 - acc
Epoch 4/5
1635/1635 [==============================] - 4s 3ms/step - loss: 0.1818 - acc
Epoch 5/5
1635/1635 [==============================] - 5s 3ms/step - loss: 0.1436 - acc
Training Accuracy: 0.9628
Testing Accuracy:  0.8653
```





Die verschiedenen Einstellungen von Hidden Layers uns Epochen zeigen, dass das Model relativ stabil ist. Allerdings sind eine Maximale Anzahl von 2 Hidden Layer in allen Sub-Modellen und unter 10 Epochen für das Training die besten Einstellungen. So erzielt das Model seine beste Genauigkeit in der Klasssifizierung und schlägt das einfache Model, das nur die Artikelüberschrift als Feature verwendet.

SVM

Support Vector Machines zur Klassifizierung des Datensets. Das folgende Vorgehen ist angelehnt an [A guide to Text Classification(NLP) using SVM and Naive Bayes with Python].

Support Vector Maschines

Neben einem Neuronalen Netz haben wir eine weitere Architektur genutzt, die Support Vector Machines. Diese beiden Ansätze sind sehr ähnlich. Trotzdem wollten wir herausfinden, ob die eine Architektur wesentlich besser ist als die andere.

Folgende Parameter haben wir dafür angepasst:

- Kernel (sigmoid, linear, rbf, poly)
- Grad des Polynoms im Kernel
- Regularization mit dem C-Wert

CODE ANZEIGEN

CODE ANZEIGEN

CODE ANZEIGEN

SVM with linear kernel

CODE ANZEIGEN

```
SVM Training Accuracy Score ->  87.33206632151517
SVM Test Accuracy Score ->  78.33333333333333
```

SVM with sigmoid kernel

CODE ANZEIGEN

```
SVM Training Accuracy Score ->  82.3513991742578
SVM Test Accuracy Score ->  77.58409785932722
```

SVM with rbf kernel

CODE ANZEIGEN

```
SVM Training Accuracy Score ->  97.45068484173275
SVM Test Accuracy Score ->  78.88379204892966
```

SVM with polynomial kernel of second degree

CODE ANZEIGEN

```
SVM Training Accuracy Score ->  98.97765253293139
SVM Test Accuracy Score ->  78.92966360856269
```

SVM with polynomial kernel of second degree with C=1.5

CODE ANZEIGEN

```
SVM Training Accuracy Score ->  99.55436136050855
SVM Test Accuracy Score ->  78.79204892966361
```

SVM with polynomial kernel of second degree with C=0.5

CODE ANZEIGEN

```
SVM Training Accuracy Score ->  96.4873189593027
SVM Test Accuracy Score ->  77.64525993883792
```

Der Versuch, statt einem Neuronalen Netzen eine Support Vector Machine zu nutzen, hat in ersten Versuchen, bei denen wir verschiedene Parametrisierung ausprobierten, kein besseres Ergebnis gebracht, auch nicht besser als das einfache Neuronale Netz mit nur einem Feature. Die Testgenauigkeit war immer unter 80%. Daher wurde diese Variante nicht mehr genauer untersucht und das Augenmerk eher auf die Neuronalen Netze mit einem und mehreren Features gelegt.

Fazit

Sarkasmus zu erkennen ist nicht leicht, weder für Maschine noch für Menschen. Viele Identifikatoren, die auf Sarkasmus hinweisen, sind nicht zwangsläufig spezielle Triggerworte, sondern vom Kontext abhängig. Der Kontext muss nicht zwangsläufig unmittelbar mit dem zu analysierenden Text aufgeklärt werden, sondern kann auch ein generelles Problem oder einen Missstand addressieren, der allegemein bekannt ist.

Ein Problem ist, dass es nur wenige Ansätze für diesen Anwendunsgfall gibt, die nicht auf Basis von Dictionaries arbeiten. Die enthaltenen *Triggerworte*, also diejenigen Worte, die eine Aussage über ein bestimmtes Feature liefern, können den ständigen Wandel einer Sprache nicht vollständig abbilden. Wortneuschöpfungen können z.B. nicht berücksichtigt werden bzw. das Wörterbuch muss ständig aktualisiert werden. Dazu kommt, dass sich ein Model möglicherweise nicht generalisieren lassen kann, weil nicht alle Sprachen gleich funktionieren. Für unterschiedliche Sprachen muss es also unterschiedliche Modelle geben, für die eine große Menge an Daten benöigt werden.

Forschern vom MIT ist es gelungen, auf Basis von Deep Learning ein Model zu entwickeln, welches Sarkasmus erkennt. Dieses arbeitet mit einer Genauigkeit von ca. 82% (Artikel). Hierbei ging es darum, Posts in sozialen Netzwerken zu klassifizieren. Die Forscher nutzen dabei auch aus, dass viele User Emojis nutzen, um bestimmte Emotionen auszudrücken. Das war aufgrund des Datensatzes in diesem Projekt nicht möglich. Stattdessen konnte jedoch hier der Artikelinhalt genauer analysiert werden.

Bei unserem Versuch haben wir zunächst nur die Überschrift analysiert und keine weiteren Features. Das ergab eine ungefähre Genauigkeit von ebenfalls 81-82%. Durch die Hinzunahme von den zuvor extrahierten Features konnte die Genauigkeit zur Klassifikation von Testdaten noch einmal auf über 86% verbessert werden. Das dieses Model sehr gut abschneidet, lässt sich dadurch erklären, dass der Kontext vermutlich sehr viel mehr Information liefert als ein einfacher Post, welcher mit Emojis versehen wurde.