

5CHIT – Marius Lassl

KINZ 24/25

Projekt: „Fake News“

21.03.2025

INHALTSVERZEICHNIS

# 1. Einleitung	2
1.1 Zielsetzung und Anforderungen	2
# 2. Datenverarbeitung	3
2.1 Einlesen und Vorbereiten des Datensatzes	3
2.2 Datensätze reduzieren	4
2.3 Textaufbereitung	5
# 3. Feature Engineering	6
3.1 Merkmalsextraktion	6
# 4. Modelltraining	7
4.1 Datenaufteilung	7
4.2 Training des Random Forest Classifiers	7
# 5. Evaluation und Tests	8
# 6. Anhang	10
6.1 Quellcode	10
6.2 Abbildungsverzeichnis	13

1. Einleitung

In diesem Projekt entwickeln wird ein System zur Erkennung von Fake News mithilfe von Natural Language Processing (NLP) und Machine Learning. Ziel ist es, Inhalte aus einem Datensatz analysieren, Merkmale zu extrahieren und ein Klassifikationsmodell (Random Forest Classifier) zu trainieren, das zwischen echten Nachrichten und Fake News unterscheiden kann.

1.1 Zielsetzung und Anforderungen

- Verständnis von NLP-Methoden wie Tokenisierung, Stoppwörter-Entfernung, POSTagging und Named Entity Recognition (NER).
- Anwendung von Machine Learning für Klassifikationsprobleme.
- Entwicklung einer praxisnahen Lösung für ein reales Problem.

Nähere Aufgabenstellungen zu den einzelnen Aufgaben befinden sich im Dokument „KI5_Projekt-FakeNews_v0.1.pdf“ und zusammengefasst in den jeweiligen Kapiteln.

Anforderungen Dokumentation:

Es soll ein Protokoll mit Deckblatt und Inhaltsverzeichnis erstellt werden, das sämtliche Schritte auf Punkt und Beistrich dokumentiert. Hierzu soll der Quellcode in das Dokument kopiert und kommentieren werden. Die Zwischenschritte/-ergebnisse sollen auch kommentiert werden, sodass maximale Transparenz gegeben ist.

HINWEIS: Es ist in den Screenshots zwecks der Übersichtlichkeit nicht immer der gesamte Code vorhanden. (Imports und sonstiger nicht unbedingt relevanter Code wurde weggelassen)

Im Anhang befindet sich dafür der komplette Quellcode für alle Teilschritte.

2. Datenverarbeitung

2.1 Einlesen und Vorbereiten des Datensatzes

- Der bereitgestellte Datensatz "news.csv" enthält Spalten für die Nachricht (Body) und ihre Klassifikation (Fake oder echt).
- Es wird ein DataFrame "data" erstellt, das nur die relevanten Spalten enthält:
 - text**: Enthält die Nachricht.
 - label**: Klassifikation (0 = echt, 1 = Fake News).

	id	url	Titel	Body	Kategorie	Datum	Quelle	Fake	Art
0	773233	http://www.der-postillon.com/2018/01/grokoleak...	Exklusiv! Das geheime WhatsApp-Chat-Protokoll...	Die Sondierungsgespräche zwischen Union und SP...	wirtschaft	2018-01-18 00:00:00	Postillon	1	NaN
1	773234	http://www.der-postillon.com/2018/01/trump-san...	Trump droht, jeden zu verspeisen, der an seine...	Nun ist es auch medizinisch offiziell bestätig...	wirtschaft	2018-01-17 00:00:00	Postillon	1	NaN
2	773235	http://www.der-postillon.com/2018/01/fdp-sondi...	Soli runter, keine Steuererhöhungen, kein Klim...	Es waren zähe Verhandlungen, doch die Freien D...	wirtschaft	2018-01-12 00:00:00	Postillon	1	NaN
3	773236	http://www.der-postillon.com/2018/01/joachim-s...	Hat sie eine Affäre? Joachim Sauer glaubt Ange...	Wo treibt sie sich immer bis spät in die Nacht...	wirtschaft	2018-01-09 00:00:00	Postillon	1	NaN
4	773237	http://www.der-postillon.com/2018/01/halb-so-s...	"Er hat ja nur HALBneger gesagt": So begründet...	Der Parteivorstand drückt nochmal ein Auge zu:...	wirtschaft	2018-01-08 00:00:00	Postillon	1	NaN

Abbildung 1: Der Datensatz news.csv [von Aufgabenstellung]

Code mit Ergebnis:

```
[65]: import pandas as pd
      filepath = "news.csv"

      #---- Load dataset ----
      def load_data(filepath):
          data = pd.read_csv(filepath, encoding='utf-8')
          data = data[['Body', 'Fake']]
          data.columns = ['text', 'label']
          return data

      data = load_data('news.csv')
      print(f"Datensatz gesamt: {len(data)}\n{data['label'].value_counts()}")

      Datensatz gesamt: 9254
      label
      0    59241
      1     4627
      Name: count, dtype: int64

[66]: data.head()
```

	text	label
0	Die Sondierungsgespräche zwischen Union und SP...	1
1	Nun ist es auch medizinisch offiziell bestätig...	1
2	Es waren zähe Verhandlungen, doch die Freien D...	1
3	Wo treibt sie sich immer bis spät in die Nacht...	1
4	Der Parteivorstand drückt nochmal ein Auge zu:...	1

Abbildung 2: Ergebnis mit Code

Erklärung Code: Die Funktion load_data liest die Datei mit den Fake News aus und gibt ein DataFrame mit den zwei Spalten „text“ und „label“ (fake oder nicht fake) zurück.

2.2 Datensätze reduzieren

Hinweis aus der Aufgabenstellung „KI5_Projekt-FakeNews_v0.1.pdf“:

Das Dataset enthält über 60.000 Nachrichten, was die Verarbeitung verlangsamt. Um die Verarbeitungszeit zu reduzieren, sollten die echten Nachrichten auf ~5.000 reduziert werden, sodass beide Klassen (echt und fake) gleich groß sind. Falls die Verarbeitung dennoch zu lange dauert, kann die Anzahl weiter verringert werden. Die Funktion `sample` kann genutzt werden, um zufällig Datensätze auszuwählen oder das DataFrame durchzumischen, insbesondere vor dem Zusammenfügen von echten und Fake News. Die gesamte Textaufbereitung erfolgt in der Funktion `preprocess_text`, die den bereinigten Text sowie extrahierte Merkmale speichert oder zurückgibt.

Datensätze reduziert:

```
[59]: #---- reduce dataset size ----
def balance_all_data():
    fake_news = data[data['label'] == 1]
    true_news = data[data['label'] == 0].sample(n=len(fake_news), random_state=42)

    combined = pd.concat([true_news, fake_news]).sample(frac=1, random_state=42).reset_index(drop=True)
    print(f"Reduzierter Datensatz gesamt: {len(combined)}\n{combined['label'].value_counts()}")
    return combined

balanced_data = balance_all_data()

Reduzierter Datensatz gesamt: 9254
label
0    4627
1    4627
Name: count, dtype: int64
```

Abbildung 5: Code zur Reduzierung der echten Nachrichten

Erklärung Code: Anzahl der wahren Nachrichten wird mit `.sample` reduziert (sollen gleich viele `true_news` wie `fake_news` vorhanden sein). `random_state` stellt sicher, dass die Zufallsprozesse reproduzierbar sind. Danach werde die Nachrichten wieder in ein DataFrame mit `concat` zusammengefügt (Indexe müssen zur Sicherheit auch wieder neu gesetzt werden; 0, 1, 2 ...).

Mit `.value_counts` kann angezeigt werden, wieviele Spalten das Label 0 und wieviele das Label 1 haben.

2.3 Textaufbereitung

- 1) Nur alphanumerische Zeichen werden berücksichtigt.
- 2) Alle Texte werden in Kleinbuchstaben umgewandelt.
- 3) Stoppwörter werden entfernt. (und Lemmatisierung)
- 4) Zusätzliche linguistische Merkmale werden extrahiert:
 - **adjective_count**: Anzahl der Adjektive (z.B. "schockierend", „groß“).
 - **adverb_count**: Anzahl der Adverbien (z.B. "offensichtlich", „extrem“).
 - **entity_count**: Anzahl der benannten Entitäten (z.B. Namen, Orte).

preprocess_text mit Ergebnis:

<pre> #----- main preprosession ----- def preprocess_text(text): #1) text = re.sub(r'^a-zA-ZäöüÄÖÜß\s',' ', text) #2) text = text.lower() #3) doc = nlp(text) #tokenisierung mit spacy tokens = [token.lemma_ for token in doc if token.text not in stop_words and not token.is_punct] #4) adjective_count = sum(1 for token in doc if token.pos_ == "ADJ") adverb_count = sum(1 for token in doc if token.pos_ == "ADV") entity_count = len(doc.ents) return " ".join(tokens), adjective_count, adverb_count, entity_count balanced_data[['text_clean', 'adjective_count', 'adverb_count', 'entity_count']] = balanced_data['text'].apply(lambda x: pd.Series(preprocess_text(str(x)))) balanced_data.head() </pre>							
		text	label	text_clean	adjective_count	adverb_count	entity_count
0		Neuartige Übertragungsmethode von Neurostimula...	0	neuartig Übertragungsmethode Neurostimulation ...	36	29	9
1		Viele hatten es schon lange geahnt, jetzt ist ...	1	viele schon lange ahnen endlich amtlich Verfas...	18	40	16
2		Der Zar lässt die Säbel rasseln: Vor dem Hinte...	1	zar Lässt säbel Rassel Hintergrund zunehmend S...	15	27	20
3		Pumpt E10 unter die Erde: Bohrturm\r\n\r\nlm Z...	1	pumpen e Erde bohrturm \r\n\r\n Zug sogenannte...	11	23	11
4		iPad 2 wird angeblich mit extrem hoher Auflösu...	0	ipad angeblich extrem hoch Auflösung vorlege...	21	53	22

Abbildung 6: Ergebnis der Textaufbereitung mit Code

Erklärung Code:

- 1) alle nicht-alphanumerische Zeichen werden mithilfe von einem Regex „gelöscht“.
- 2) alle Texte werden mit lower() in Kleinbuchstaben umgewandelt
- 3) zuerst Tokenisierung mit spacy, danach werden Stoppwörter und sonstige Sonderzeichen (eigentlich nicht notwendig, Regex sorgt bereits dafür dass keine Sonderzeichen dabei sind) rausgefiltert und der Rest lemmatisiert
- 4) Adjektive, Adverbien und Entitäten der Tokens werden gezählt

Mithilfe eines Lambda-Ausdrucks werden diese Schritte auf alle Spalten angewendet.

3. Feature Engineering

3.1 Merkmalsextraktion

- Texte werden in numerische Werte umgewandelt, da Machine Learning-Modelle nur mit Zahlen arbeiten können.
- Verwendung von CountVectorizer, um ein Vokabular aller einzigartigen Wörter zu erstellen und ihre Häufigkeit in den Texten zu erfassen.
- Optionale Reduktion der Merkmale durch Begrenzung der Wortanzahl (max_features).

```
import numpy as np

def create_feature_df(vectorizer):
    X_text = vectorizer.fit_transform(balanced_data['text_clean']).toarray()

    X_text_df = pd.DataFrame(X_text, columns=vectorizer.get_feature_names_out())
    X_other_df = balanced_data[['adjective_count', 'adverb_count', 'entity_count']].reset_index()

    return pd.concat([X_text_df, X_other_df], axis=1)

# ----- feature extraction -----
vectorizer = CountVectorizer(max_features=1000)
X = create_feature_df(vectorizer)
y = balanced_data['label']
```

X

	überhaupt	übernehmen	überraschend	überzeugen	adjective_count	adverb_count	entity_count
0	0	0	0	0	36	29	9
1	0	0	0	0	18	40	16
0	0	0	0	0	15	27	20
0	0	0	0	0	11	23	11
0	0	0	0	0	21	53	22
...
0	0	0	0	1	16	30	11

Abbildung 7: Code zur Merkmalsextraktion

In der neuesten Version des Projekts ist die Wortanzahl auf 1000 begrenzt; mit einer Begrenzung von 5000 konnte die Genauigkeit nur um 1% verbessert werden.

Erklärung Code: In der create_feature_df Funktion wird mithilfe eines Vektorizer die ganzen Wörter (wegen max_feats=1000 nur die 1000 häufigsten) gezählt und als Array gespeichert. Daraus wird danach ein Dataframe mit den dementsprechenden Spaltennamen erstellt. Ein zweites Dataframe wird aus den zusätzlichen Features erstellt (adj, adv, entities) und diese Dataframes werden zum Schluss zusammengeführt mit concat und X nimmt den Wert dieses neuen Dataframes an. Die Labels (0 oder 1) werden als y gespeichert.

4. Modelltraining

4.1 Datenaufteilung

- Der vorbereitete Datensatz wird in Trainings- (80%) und Testdaten (20%) aufgeteilt.

4.2 Training des Random Forest Classifiers

- Der Random Forest Classifier wird trainiert, um Nachrichten zu klassifizieren.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# ----- Modelltraining -----
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

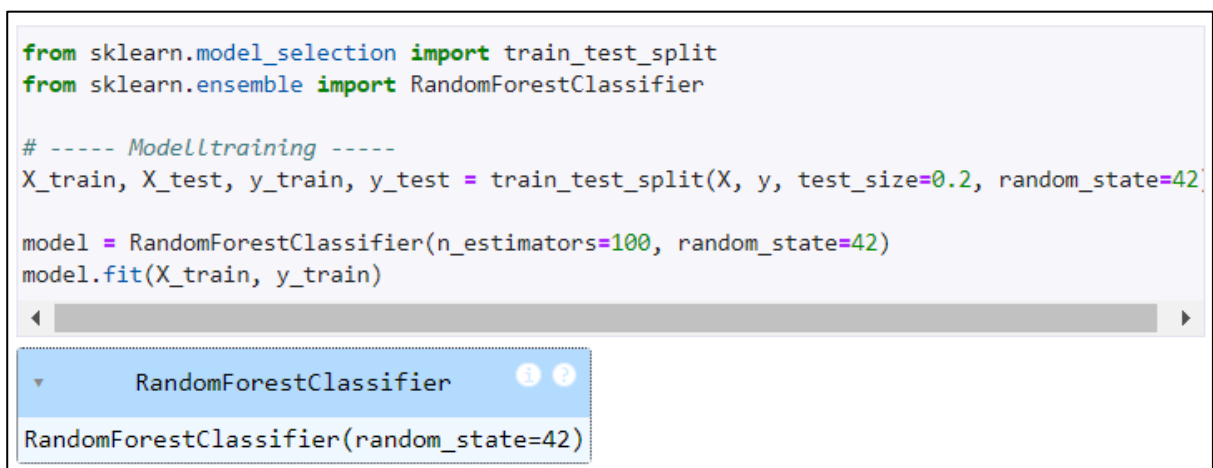


Abbildung 8: Code Modelltraining

Erklärung Code: mit `train_test_split` werden nun `X` und `y` aufgeteilt. `test_size = 0.2` gibt dabei an, dass 20% für Testdaten verwendet werden und 80% für Trainingsdaten. `random_state` stellt wieder fest, dass man bei einem erneuten Durchlauf das gleiche Ergebnis erzielt.

Danach wird ein Model definiert. Hierfür wird ein `RandomForestClassifier` verwendet, `random_sate` ist wieder festgelegt und es sollen 100 einzelne Bäume erstellt werden. Mehr Bäume können Genauigkeit verbessern, brauchen dafür aber auch mehr Rechenleistung.

Zum Schluss wird mit `fit(X_train, y_train)` das eigentliche Model mit den Trainingsdaten trainiert.

5. Evaluation und Tests

- Berechnung der Modellgenauigkeit.
- Untersuchung, wie gut echte Nachrichten von Fake News unterschieden werden.

```
from sklearn.metrics import accuracy_score
import joblib

# ----- Genauigkeit -----
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Genauigkeit: {accuracy:.2f}")

# ----- Modell speichern -----
joblib.dump(model, "model/fake_news_model.pkl")
joblib.dump(vectorizer, "model/vectorizer.pkl")

print("Modell gespeichert!")

Genauigkeit: 0.96
Modell gespeichert!
```

Abbildung 9: Genauigkeit des Models anzeigen; Modell speichern

Erklärung Code: Mit `predict(X_test)` wird dem Model nun die Testdaten übergeben. Dieses klassifiziert die Daten und dann können die vorausgesagten Labels (`y_pred`) mit den eigentlichen Labels (`y_test`) verglichen werden. Daraus ergibt sich die Genauigkeit des Models (hier wurden 96% der Testdaten richtig erkannt).

Abschließend wird das Modell und der Vektorizer noch zusätzlich gespeichert.

Abschließend wurde das Model wieder geladen und jeweils drei echte und gefälschte Test-Nachrichten klassifiziert.

```
def predict_news(text):
    text_cleaned, adj, adv, ent = preprocess_text(text)

    text_vectorized = vectorizer.transform([text_cleaned]).toarray()
    text_df = pd.DataFrame(text_vectorized, columns=vectorizer.get_feature_names_out())
    features_df = pd.DataFrame([adj, adv, ent], columns=['adjective_count', 'adverb_count', 'entity_count'])
    combined = pd.concat([text_df, features_df], axis=1)

    prediction = model.predict(combined)[0]
    return "Fake" if prediction == 1 else "Echt"

results = [(text, predict_news(text)) for text in test_news]
for text, prediction in results:

    if prediction == "Echt":
        print(f"{green_c}{prediction}{black_c} {text}")
    else:
        print(f"{red_c}{prediction}{black_c} {text}")

[Echt] GPS.AT übernimmt Grazer Unternehmen XLOC Tracking Systems und setzt Wachstumskurs auch 2015 fort
[Echt] TechFirma übernimmt führendes Unternehmen im Bereich Künstliche Intelligenz und setzt globales Expansionsziel 2025 fort.
[Echt] EnergieRiese investiert massiv in erneuerbare Technologien und plant Expansion in den asiatischen Markt.
[Fake] Schockierender Bericht: Politiker kontrollieren das Wetter!
[Fake] Prominente warnen: Die Erde ist in Wahrheit flach!
[Fake] Wissenschaftler entdecken Heilmittel gegen Krebs – aber Pharmaindustrie blockiert es!
```

Abbildung 10: Klassifizierung mit Test-Nachrichten

Erklärung Code: Es würden Test-Nachrichten mit ChatGPT generiert und getestet, ob das trainierte Modell diese richtig erkennt. In der Funktion predict_news wird wieder ein Dataframe mit den Features erstellt (Optimierungspotential?) und anschließend klassifiziert das Model die Test-Nachrichten.

results ist ein Array mit den Nachrichten und der zusätzlichen Information, ob diese „Fake“ oder „Echt“ sind. Zum Schluss wird results noch farbig ausgegeben.

6. Anhang

6.1 Quellcode

```
import pandas as pd
filepath = "news.csv"

#----- Load dataset -----
def load_data(filepath):
    data = pd.read_csv(filepath, encoding='utf-8')
    data = data[['Body', 'Fake']]
    data.columns = ['text', 'label']
    return data

data = load_data('news.csv')
print(f"Datensatz gesamt: {len(balanced_data)}\n{data['label'].value_counts()}")
```

```
#----- reduce dataset size -----
def balance_all_data():
    fake_news = data[data['label'] == 1]
    true_news = data[data['label'] == 0].sample(n=len(fake_news), random_state=42)

    combined = pd.concat([true_news, fake_news]).sample(frac=1,
random_state=42).reset_index(drop=True)
    print(f"Reduzierter Datensatz gesamt:
{len(combined)}\n{combined['label'].value_counts()}")
    return combined

balanced_data = balance_all_data()
```

```
import pandas as pd
import re
import nltk
import spacy
from nltk.corpus import stopwords

#----- configs -----
nlp = spacy.load("de_core_news_sm")

#nltk.download('stopwords')
stop_words = set(stopwords.words('german'))

#----- main preproccession -----
def preprocess_text(text):
    #1)
    text = re.sub(r'^a-zA-ZäöüÄÖÜß\s', '', text)

    #2)
    text = text.lower()

    #3)
    doc = nlp(text) #tokenisierung mit spacy
    tokens = [token.lemma_ for token in doc if token.text not in stop_words and not
token.is_punct]
```

```
#4)
adjective_count = sum(1 for token in doc if token.pos_ == "ADJ")
adverb_count = sum(1 for token in doc if token.pos_ == "ADV")
entity_count = len(doc.ents)

return " ".join(tokens), adjective_count, adverb_count, entity_count

balanced_data[['text_clean', 'adjective_count', 'adverb_count', 'entity_count']] =
balanced_data['text'].apply(
    lambda x: pd.Series(preprocess_text(str(x)))
)

balanced_data.head()
```

```
import numpy as np

def create_feature_df(vectorizer):
    X_text = vectorizer.fit_transform(balanced_data['text_clean']).toarray()

    X_text_df = pd.DataFrame(X_text, columns=vectorizer.get_feature_names_out())
    X_other_df = balanced_data[['adjective_count', 'adverb_count',
                                'entity_count']].reset_index(drop=True)

    return pd.concat([X_text_df, X_other_df], axis=1)

# ----- feature extraction -----
vectorizer = CountVectorizer(max_features=1000)
X = create_feature_df(vectorizer)
y = balanced_data['label']
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# ----- Modelltraining -----
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
from sklearn.metrics import accuracy_score
import joblib

# ----- Genauigkeit -----
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Genauigkeit: {accuracy:.2f}")

# ----- Modell speichern -----
joblib.dump(model, "model/fake_news_model.pkl")
joblib.dump(vectorizer, "model/vectorizer.pkl")

print("Modell gespeichert!")
```

```
import joblib
import numpy as np
import pandas as pd

red_c = "\033[91m["
green_c = "\033[92m["
black_c = "]\033[0m"

# ----- Modell Laden -----
model = joblib.load("model/fake_news_model.pkl")
vectorizer = joblib.load("model/vectorizer.pkl")

test_news = [
    #3 echte Nachrichten
    "GPS.AT übernimmt Grazer Unternehmen XLOC Tracking Systems und setzt Wachstumskurs auch 2015 fort",
    "TechFirma übernimmt führendes Unternehmen im Bereich Künstliche Intelligenz und setzt globales Expansionsziel 2025 fort.",
    "EnergieRiese investiert massiv in erneuerbare Technologien und plant Expansion in den asiatischen Markt.",
    #3 gefälschte Nachrichten
    "Schockierender Bericht: Politiker kontrollieren das Wetter!",
    "Prominente warnen: Die Erde ist in Wahrheit flach!",
    "Wissenschaftler entdecken Heilmittel gegen Krebs – aber Pharmaindustrie blockiert es!",
]

# ----- Fake-News-Klassifikation -----
def predict_news(text):
    text_cleaned, adj, adv, ent = preprocess_text(text)

    text_vectorized = vectorizer.transform([text_cleaned]).toarray()
    text_df = pd.DataFrame(text_vectorized, columns=vectorizer.get_feature_names_out())
    features_df = pd.DataFrame([[adj, adv, ent]], columns=['adjective_count', 'adverb_count', 'entity_count'])
    combined = pd.concat([text_df, features_df], axis=1)

    prediction = model.predict(combined)[0]
    return "Fake" if prediction == 1 else "Echt"

results = [(text, predict_news(text)) for text in test_news]
for text, prediction in results:

    if prediction == "Echt":
        print(f"{green_c}{prediction}{black_c} {text}")
    else:
        print(f"{red_c}{prediction}{black_c} {text}")
```

6.2 Abbildungsverzeichnis

Abbildung 1: Der Datensatz news.csv [von Aufgabenstellung].....	3
Abbildung 2: Ergebnis mit Code	3
Abbildung 3: Code zur Reduzierung der echten Nachrichten	4
Abbildung 4: Ergebnis der Textaufbereitung mit Code.....	5
Abbildung 5: Code zur Merkmalsextraktion.....	6
Abbildung 6: Code Modelltraining.....	7
Abbildung 7: Genauigkeit des Models anzeigen; Modell speichern.....	8