

# Klassifizierung von Katastrophen- Tweets mit NLP

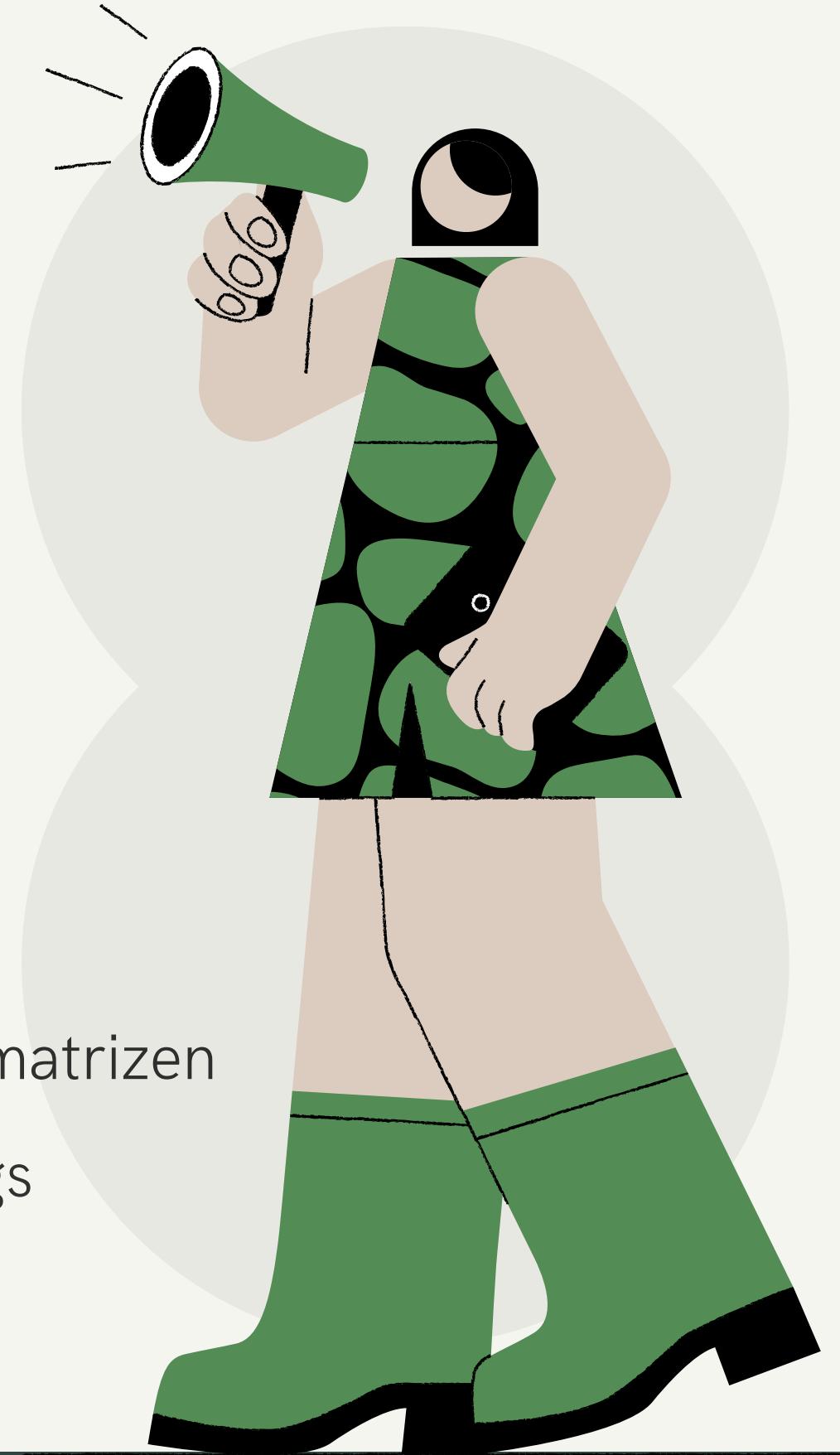
Ein Machine-Learning-Projekt mit TF-IDF &  
Klassifikationsmodellen



# Agenda

Was uns heute erwartet:

- 🐦 Problem & Motivation
- 🐦 Datensatz im Überblick
- 🐦 Textvorverarbeitung (NLP)
- 🐦 TF-IDF Feature Engineering
- 🐦 Modelltraining
- 🐦 Modellvergleich & Konfusionsmatrizen
- 🐦 Herausforderungen & Learnings
- 🐦 Finales Modell & Ausblick





# Problem & Motivation

## Problemstellung:

Nicht jeder dramatische Tweet beschreibt eine echte Katastrophe – aber in Krisensituationen zählt jede Minute.

## Ziel des Projekts:

👉 Ein Modell bauen, das automatisch erkennt, ob ein Tweet  
1 = reale Katastrophe oder 0 = keine Katastrophe beschreibt.

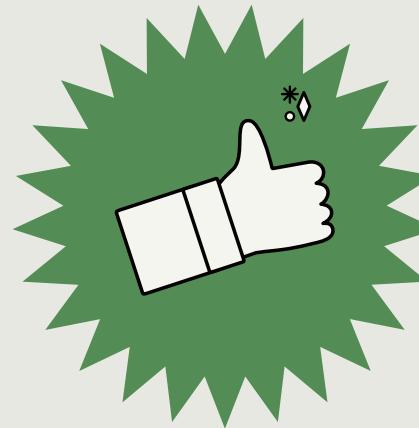
## Warum ist das wichtig?

- Unterstützung für Krisenkommunikation & Einsatzkräfte
- Filtern von Rauschen & Übertreibung in Social Media
- Grundlage für automatisierte Monitoring-Systeme

## Beispiel-Tweets:

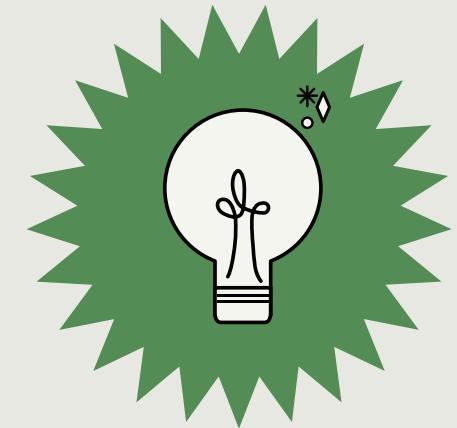
- "Massive explosion downtown... buildings collapsed." → 1 (Katastrophe)
- "My phone just exploded with messages 😂" → 0 (keine Katastrophe)

# Datensatz im Überblick



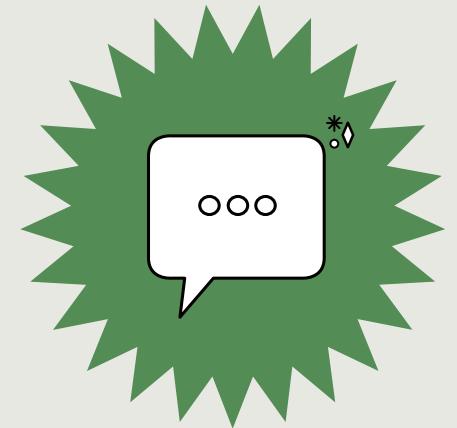
## Spalten:

- text → der eigentliche Tweet
- target → 1 = Katastrophe, 0 = keine Katastrophe



## Klassenverteilung:

- 0 (keine Katastrophe): 4.342 Tweets
- 1 (Katastrophe): 3.271 Tweets



## Interpretation:

- Mini-Balkendiagramm (0 vs. 1)
- kleiner Ausschnitt (2-3 Zeilen) aus der CSV als Screenshot



# Textvorverarbeitung (NLP)

**Ziel:** Chaotische Tweets in saubere, modellierbare Texte verwandeln.

## Preprocessing-Schritte:

1. Kleinbuchstaben
2. Entfernen von Satzzeichen & Sonderzeichen
3. Tokenisierung (Wörter trennen)
4. Entfernen englischer Stoppwörter
5. Lemmatisierung (running → run)
6. Tokens wieder zu einem String zusammenfügen

## Beispiel:

### Original:

"Our Deeds are the Reason of this #earthquake ..."

### Bereinigt:

"deed reason earthquake may allah forgive u"

## Kernaussage:

- ➔ Weniger Rauschen, mehr Fokus auf bedeutungsstarke Begriffe.
- ➔ Legt die Basis dafür, dass das Modell überhaupt sinnvolle Muster lernen kann.

# Feature Engineering mit TF-IDF

## Warum TF-IDF?

“

- Gewichtet Wörter nach Wichtigkeit im Kontext
- Häufige Füllwörter bekommen wenig Gewicht
- Relevante Begriffe wie “earthquake”, “fire”, “flood” werden hervorgehoben

## Setup:

“

- `TfidfVectorizer(max_features = 5.000)`
- Ergebnis: Sparse-Matrix mit
  - → 7.613 Zeilen (Tweets)
  - → 5.000 Spalten (Features/Wörter)

## Kernaussage:

“

- ➡ Aus rohem Text wird eine numerische Repräsentation, auf der klassische ML-Modelle arbeiten können.

# Modelltraining

Ich habe drei Modelle trainiert und verglichen:

1. Logistische Regression (Baseline)
2. Random Forest Classifier
3. MLP Neuronales Netz  
(MLPClassifier)

## Gemeinsame Basis:

- Gleiche TF-IDF-Features (5.000)
- Gleicher Train/Test-Split (80/20, stratify)

## Ergebnisse (Accuracy auf Testset):

- Logistische Regression: 81,88 %
- MLP Neuronales Netz: 81,68 %
- Random Forest: 80,43 %

## Kernaussage:

→ Alle drei Modelle liegen relativ nah beieinander – aber sie machen unterschiedliche Fehlerarten.

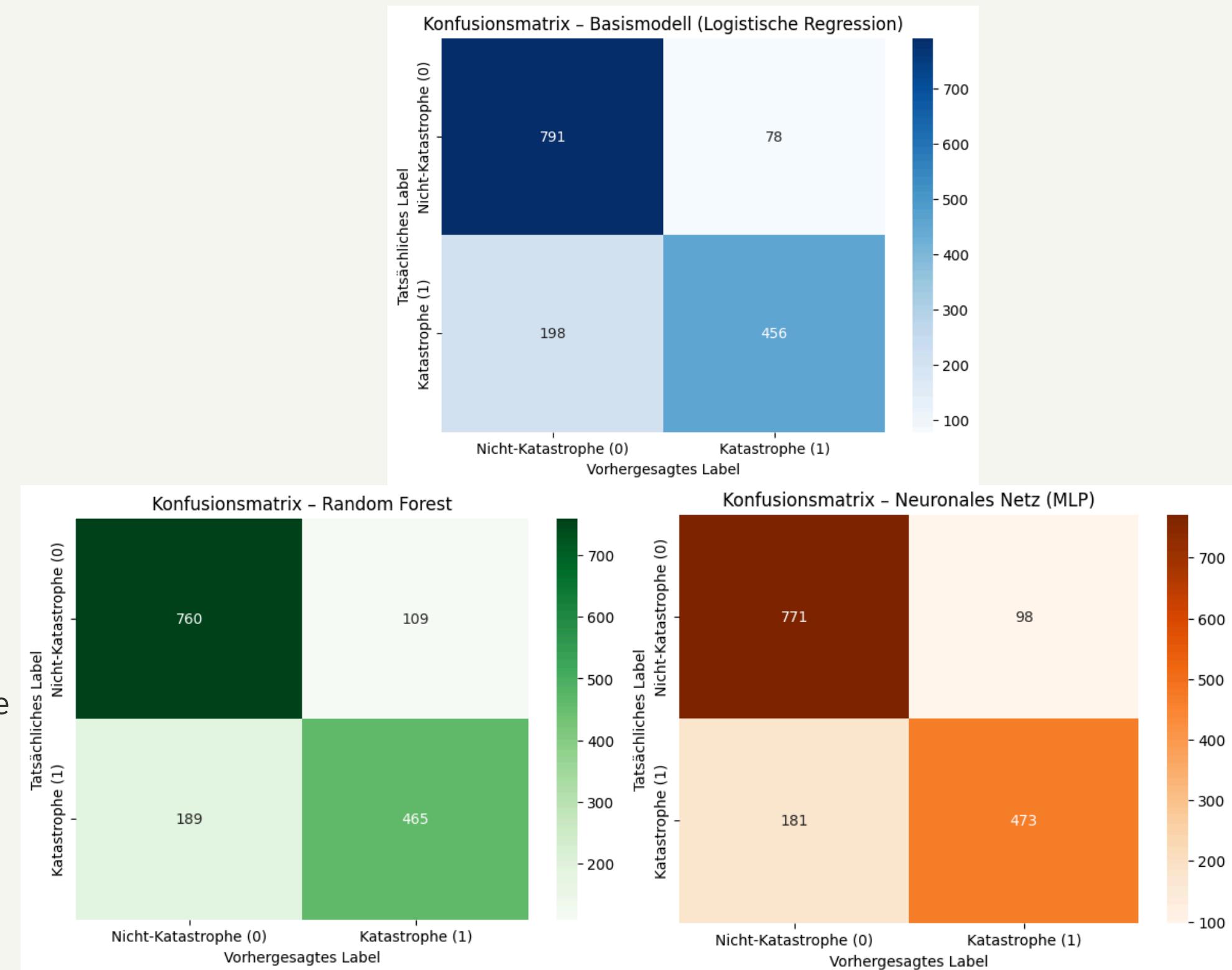
# Modellvergleich & Konfusionsmatrizen

## Was ich mir anschauе:

- Wie viele Katastrophen-Tweets werden übersehen? (False Negatives)
- Wie viele Fehlalarme produziert das Modell? (False Positives)

## Kurzvergleich:

- **Logistische Regression (blau)**
  - Gute Balance
  - FN = 198, FP = 78
- **Random Forest (grün)**
  - Erkennt etwas mehr Katastrophen (höherer Recall für Klasse 1)
  - Dafür mehr Fehlalarme (FP = 109)
- **MLP Neuronales Netz (orange)**
  - Beste Gesamt-Accuracy nach LR
  - FN = 181 (weniger übersehene Katastrophen als LR)
  - FP = 98



# “ Herausforderungen & wichtigste Learnings

## Herausforderungen:

- Tweets sind kurz, ironisch, voller Emojis & Hashtags
- Leicht unausgeglichene Klassen → Risiko für False Negatives
- Hochdimensionale TF-IDF-Matrix (5.000 Features)

## Was ich gelernt habe:

- Gute Vorverarbeitung ist entscheidend für die Modellqualität
- TF-IDF + Logistische Regression ist ein sehr starker Standard für Textklassifikation
- Accuracy allein reicht nicht – Konfusionsmatrix, Recall & F1-Score sind wichtig
- Cross-Validation (~70 % Accuracy im Schnitt) gibt ein realistischeres Bild als ein einzelner Train/Test-Split



# Finales Modell & Ausblick

Finale Entscheidung:



Logistische Regression als bestes Modell

## Warum?

- Höchste Accuracy (81,88 %)
- Sehr stabile, robuste Performance
- Einfach zu trainieren und gut interpretierbar
- Vergleichbare Ergebnisse zum MLP, aber weniger komplex

## Mögliche nächste Schritte:

- Nutzung von Word Embeddings (Word2Vec, GloVe)
- Einsatz eines BERT-Modells für kontextsensitives Verstehen
- Feintuning des MLP & Class Weights für Klasse 1
- Kleine API / Demo-App, die Tweets live klassifiziert

## Persönliches Fazit:

Dieses Projekt hat mir den kompletten NLP-Workflow gezeigt, von rohem Text bis zum ausgewählten Modell, und mir ein viel besseres Verständnis für die Wirkung von Preprocessing, Feature Engineering und Modellvergleich gegeben.



# Danke für die Aufmerksamkeit

- alle Infos und detaillierten Informationen findet ihr auf meinem [GitHub](#)
- Vielen Dank fürs Zuhören!