Deviations from the original plan

This is very important to clarify in the beginning.

- I realized that many news outlets do not allow web scrapers. For some sites the scraper would not work. I therefore had to reduce my list of outlets. However, I still managed to diversify. I have far right and left news, state news, mainstream media and not so very much known media.
- I realized during the labelling, that the labels were too many and too complicated. As I initially chose one final algorithm from a github repo, that had the labels “Fake News”, “Credible”, “Extreme Bias” and “Clickbait”, I also reduced them to this.
- I did not clarify this, but in this dataset, the news can have multiple labels. This is the most realistic approach, but also complicated things.
- This leads me to the automatic labelling. There was no “well-established model” that could immediately label my data. The issue was the possibility of multiple labels. I therefore made my own model.
- The multi-label issue continues with the deep learning algorithm part. I also had to make my own simple model.

Error Metric and Target

1. Automatic Labelling of the Dataset

For the automatic labelling task, I chose precision as the primary error metric. The reason for this is that, in fake news detection, false positives can have a larger impact than false negatives. It's more important to label correctly (precision) rather than label extensively but incorrectly. My target for this step was a precision of 60%.

Even in manual labelling, assigning correct labels was challenging due to subjectivity and ambiguity, so I felt this was a reasonable baseline for an automated process. A higher threshold (e.g., 80% or above) would likely require a level of dataset refinement and fine-tuning that was beyond the scope of this task.

2. Deep Learning Fake News Detection Algorithm

For the deep learning task, the primary error metric was accuracy. My goal was to achieve at least 60% accuracy, which I considered a fair baseline for the following reasons:

I intentionally kept the model simpler with less fine-tuning to focus on experimenting with my dataset rather than achieving state-of-the-art performance.

The dataset was self-created, which adds noise and variance compared to large, professionally curated datasets often used in benchmarks.

Achieving a higher level of accuracy would likely require hyperparameter optimization, feature engineering, or additional preprocessing, which I deprioritized in favor of focusing on the main project tasks and the limited time.

To benchmark my results, I referred to the Fake News Detection GitHub Repository (https://github.com/AIRLegend/fakenews), which claims the following accuracies for their models on different datasets:

LSTM-based architecture: 91% accuracy (TI-CNN) | 76% accuracy (FNC)

CNN-based architecture: 97% accuracy (TI-CNN) | 82% accuracy (FNC)

BERT-based architecture: 97% accuracy (TI-CNN) | 76% accuracy (FNC)

While these benchmarks demonstrate high performance, I understood that replicating them exactly would not be realistic for this task, given differences in dataset, architecture, and computational resources.

Also, after further searching, I was not able to determine, whether this is for binary prediction or for the four class labels.

Reasoning Behind Chosen Metrics

Precision over Recall: For fake news detection, false positives (classifying true news as fake) could undermine user trust in the system, so precision is more critical.

Accuracy: Provides a straightforward measure of overall performance and is commonly reported in literature, making it a useful point of comparison.

By prioritizing interpretability and focusing on achieving a reasonable baseline, I aimed to balance the complexity of the project with practical time and resource constraints.

Achieved Metrics

For this part, I tried out with a multi-label classifier approach with 5-fold cross-validation and the following classifiers: RandomForestClassifier, LogisticRegression, XGBClassifier, LGBMClassifier

The model I finally chose resulted in the following metrics for the test set:

Classification Report:

	precision	recall	f1-score	support
0	0.75	0.04	0.08	75
1	0.59	0.16	0.26	98
2	0.65	0.21	0.31	96
3	0.71	0.92	0.80	206
micro avg	0.69	0.48	0.57	475
macro avg	0.67	0.33	0.36	475
weighted avg	0.68	0.48	0.48	475
samples avg	0.68	0.64	0.65	475

Which meant, that I reached my threshold precision everywhere but for one label.

Now for the deep learning algorithm I tried 3 different variations:

Full Data Approach

This is where I naively just split the whole data (manually labelled and automatically) and train a small NN:

Test Accuracy= 87.0%, Test Loss= 0.18

	precision	recall	f1-score	support
0	0.00	0.00	0.00	41
1	0.50	0.01	0.02	87

```
           2       0.75      0.04      0.08        69
           3       0.91      1.00      0.95       972

   micro avg       0.91      0.83      0.87      1169
   macro avg       0.54      0.26      0.26      1169
weighted avg       0.84      0.83      0.80      1169
 samples avg       0.89      0.88      0.88      1169
```

We can see a nice test accuracy of 87 %. We can also see, that one label does not get detected.

Pre-training on automated data, fine-tuning on manually labelled data

Test Loss: 67.24 %, Test Accuracy: 86.16 %

```
Classification Report:
              precision    recall  f1-score   support

   Fake News       0.00      0.00      0.00        36
Extreme bias       0.00      0.00      0.00        81
   clickbait       0.00      0.00      0.00        73
    credible       0.89      1.00      0.94       970

   micro avg       0.89      0.84      0.86      1160
   macro avg       0.22      0.25      0.24      1160
weighted avg       0.74      0.84      0.79      1160
 samples avg       0.89      0.88      0.88      1160
```

Surprisingly this approach was worse, as the model stagnated. It clearly favored one label.

2-label approach

During the time I spend with the dataset, I realized that the four labels might be too complicated. I reduced them to credible and not credible (which includes all three kind of fake news labels)

Test Accuracy= 91.0 %, Test Loss= 21.0 %

```
              precision    recall  f1-score   support

           0       1.00      0.04      0.08        76
           1       0.91      1.00      0.95       923

   micro avg       0.91      0.93      0.92       999
   macro avg       0.96      0.52      0.51       999
weighted avg       0.92      0.93      0.89       999
 samples avg       0.91      0.91      0.91       999
```

We have a high precision here.

Further analysis will be done on the actual report. This is just to showcase the results.

Time spent:

Creating crawlers and running/debugging them: 35h

Reading the articles and manually labelling them: 30h

Data analysis pipelines: 5h

Labelling approaches: 20h

Implementation of deep learning algorithm: 10h

Testing, cleaning up, some explanations and this document: 5h

Additional Information

Python 3.12.0 was used

To run the crawlers:

- Go to the directory where the scrapy.cfg file is located.
- To run: “scrapy crawl xxx” (exchange xxx for the name of the crawler that is specified in the code e.g. “bild”)

Testing that was done:

- Testing the model architecture (that all layers are there and the output shape)
- Testing the forward pass (check of shapes and probabilities)
- Testing that the training ran and the loss decreased.
- Testing the multi-label classifier shapes and probabilities

The testing was done on mock data

I understand, that testing via the python module is best practice. Unfortunately I must admit, that I completely forgot about this part until 16.12., when I did not really have the time anymore to implement it this way.