


---

# FAKE NEWS PROJECT - GROUP 5

---

## TECHNICAL REPORT

 **Mateo Anusic**

Data Science

Datalogisk Institut, Københavns Universitet  
djz709@alumni.ku.dk

 **Emil Thorlund**

Data Science

Datalogisk Institut, Københavns Universitet  
gpv679@alumni.ku.dk

 **Lucas A. Rosing**

Data Science

Datalogisk Institut, Københavns Universitet  
qgc922@alumni.ku.dk

 **Victor V. Bergien**

Data Science

Datalogisk Institut, Københavns Universitet  
tcg780@alumni.ku.dk

March 31, 2023

## 1 Data Processing

We are working with a subset of the FakeNewsCorpus dataset, which can be retrieved from the following source: [https://raw.githubusercontent.com/several27/FakeNewsCorpus/master/news\\_sample.csv](https://raw.githubusercontent.com/several27/FakeNewsCorpus/master/news_sample.csv). Our analysis involves a series of pre-processing steps to effectively clean and prepare the text data for further exploration.

**Text cleaning and tokenization:** We begin by cleaning the content of the articles using a combination of the NLTK library's tokenization function and custom regular expressions. This process involves breaking the text into individual words which we represent as tokens, while also removing unwanted elements such as punctuation marks, special characters, and other irrelevant textual components.

**Stopword removal:** Afterwhich, we eliminate common words or stopwords, which do not carry significant meaning and can be found across various documents, using the `nltk.corpus` module. By removing stopwords, we can focus on the more meaningful and informative words that are specific to each article, thus enhancing the efficiency of our analysis. Prior to remove the stopwords from our data, our token/frequency plot initially bared a stark resemblance to a plot of Zipf's law, after removing the undesired stopwords, our plot appeared much flatter and consistant. After creating this plot, we decided to use a much more overall plot, only showing the reduction in total amount of unique words.

The efficacy of our stopwords removal can be viewed below, as the output from print statements found in `Part_1_post.ipynb`:

```
Unique words before preprocessing: 19293
Unique words after removing stopwords: 17737
Reduction in percentage: 8.07%
```

**Stemming:** Lastly, we perform stemming using the `nltk.stem` module, which involves reducing words to their root or base form. This step is crucial in consolidating different variations of a word into a single term, thereby enabling more accurate comparisons and identification of similar content across articles

Similar to our stopwords removal, please refer to the output from our print statements found in `Part_1_post.ipynb` for an example of the result of stemming our subset:

```
Unique words after stemming: 12632
Reduction in percentage: 34.03%
Unique words after stemming & removing stopwords: 12632
```

We chose to represent the FakeNewsCorpus dataset using a list of dictionaries in Python, where each dictionary corresponds to a row in the dataset, and the keys represent the column names. This representation is memory-efficient and allows for easy manipulation as well as filtering of data based on specific criteria. Additionally, this format can be seamlessly converted into a pandas `dataFrame` if needed, which provides more advanced data manipulation and analysis capabilities. which we changed to pandas later downstream.

**Inherent problems with the data:** Tokenization and text cleaning: The 'content' column contained a mix of words, punctuation marks, special characters, numbers, dates, and URLs, which needed to be cleaned and tokenized for meaningful analysis.

**Missing or inconsistent values:** Some columns contained missing values or 'None', which required handling to ensure the integrity of the dataset and avoid potential errors in downstream analyses.

**Column selection:** The dataset contained numerous columns, some of which were not relevant to our analysis. We had to identify and select the most relevant columns ('type', 'content', and 'title') for further processing.

Counting the number of URLs in the content, counting the number of dates in the content, and counting the number of numeric values in the content can be seen below,

```
Number of <NUM> tokens: 2487
Number of <DATE> tokens: 40
Number of <URL> tokens: 329
```

Please refer to the output below for a calculation of the frequency of each article type in the 'type' column:

fake	155
conspiracy	31
political	23
NaN	12
unreliable	6
bias	6
junksci	6
unknown	6
reliable	3
clickbait	1
hate	1

If we generalize it to the bigger FakeNewsCorpus we can conclude the following: The dataset is imbalanced: The 'fake' category has the highest number of articles (155) compared to other types. This imbalance may lead to a biased model that is more accurate in identifying 'fake' articles while performing poorly on other types.

We worked with a subset of the FakeNewsCorpus dataset, approximately 3-4 GB in size. We discovered that the initial 1.5 million rows exhibited a high degree of class imbalance. This imbalance could potentially lead to biased model training and poor generalization to unseen data. To address this issue, we adopted a more robust data preparation strategy that involved shuffling the data extraction process. This enabled us to obtain a more balanced and representative sample of the entire dataset, which in turn facilitated the development of a more accurate and reliable Fake News prediction model. Our primary focus is on the 'type', 'content', and 'title' columns, as they provide the most relevant information for our analysis. We clean the data by removing rows with missing values and those containing 'None'. Subsequently, we convert the 'type' column into a binary format where "reliable" articles are labeled as 1 and all other articles as 0. This is done for the binary classification model. In the case of a multi-class model, we assign integer labels to each article type, such as "fake" = 0, "reliable" = 1, "rumor" = 2, and so on, to facilitate a more detailed classification. This allows for a broader understanding of the data, as it can help identify specific types or characteristics of the articles rather than just a binary classification.

Please refer to Figure 1. "Part 1, Task 1" for visual representation.

## 2 Simple Model

During the process of creating our simple model, the first step we took was determining how we would group and represent the different labels present in the FakeNewsCorpus dataset. This required some thought and justification as we would be manually selecting which labels would be seen as reliable or fake for our simple binary classifier. Ultimately, we decided that all articles that were not labeled as 'reliable' would be seen as 'fake'. It is worth noting that this does represent the dataset as being quite unbalanced, as only 1,920,139 articles or roughly 20.4% of the dataset was labeled as 'reliable'. It is also worth noting that were we to have labeled 'political' articles as reliable our subset would have been more balanced as a result.

Once we decided how we would label our data so that it would be compatible with a binary model, our next step was to choose what model to use. Immediately, we opted to use a logistic regression model as it excels in binary classification. Though we had also attempted to use a two linear models, one trained on fake news and the other on reliable articles. Unfortunately though, this endeavour did not prove to be fruitful, as we found that a linear model is fundamentally incompatible with our dataset. As such, we proceeded with only our logistic regression model.

As we chosen to continue with our logistic regression model, it follows that our next step was to give our model the best chance at success. This was done by choosing relevant parameters. Firstly, we to ensure consistency and reproducibility of our results, a random state of 0 was chosen. The argument `class_weight = 'balanced'` was used as it was previously stated that the FakeNewsCorpus dataset is quite unbalanced with regards to 'reliable' articles.

Furthermore, in order to ensure that our simple model performs optimally, we chose to tune the logistic regression models hyperparameter `C` to maximize the models accuracy. This was done by creating a `for`-loop to iterate through a list containing different values for `C` and identifying which yielded the highest accuracy. It is worth noting that we also explored the option of tuning the hyperparameter `C` with regards to the `f1-score`. However, we ultimately decided to rely on the accuracy instead as we deemed the `f1-score` to be sufficient.

Additionally, we did not find that incorporating the use of meta-data in our model made sense for the sake

of fake news prediction as not all rows have such meta-data as well as it's usefulness was overshadowed by the sheer amount of data contained within the 'content' column which we initially focused on.

Lastly, we believe it should be mentioned, that in regards to confirming whether an article is reliable or fake, including other metadata features in addition to the content and type of the articles could actually provide valuable information to the model and potentially improve its performance. Metadata such as author(s) of the article can provide information about the credibility and expertise of the author, and can be used to distinguish between articles written by reputable and less reputable authors. Perhaps even Publication Date, can be useful in confirming the reliability of an article because it can be used to detect trends and changes over time, and to identify articles that are outdated or no longer relevant. And lastly, it could be considered whether the length of the article could tilt the document to "reliable" or "fake", because it can indicate the depth and quality of the article, and can help distinguish between more in-depth and superficial articles. Domain's, may also hold some influence in determining an articles legitimacy. Consider, an article published by The Onion, as this is a satirical website our model should be able to identify a correlation between an article being published by The Onion and it's label and thus deem it as fake, given the domains significance in this case.

### 3 Advanced Model

We developed a Fake News predictor using the Complement Naive Bayes classifier. First, the feature and label data are read from two separate CSV files. Afterwhich, the title and content features are vectorized with CountVectorizer using bigrams, and the resulting sparse matrices are stored in distinct files. Subsequently, the sparse matrices for the title and content are merged. This model was designed to work as a multi-class classifier, so that it is capable of identifying each type of label present in the FakeNewsCorpus dataset. This is as opposed to using a simpler binary model as with our logistic regression model. Using this approach we are better able to gauge the types of articles contained in the dataset, as well as the models performance.

Bigrams and sparse matrices were utilized in part 3 to capture contextual information between words, enhance model understanding of textual nuances and detect patterns associated with fake news. Sparse matrices efficiently represented the high-dimensional feature space introduced by bigrams, reducing memory requirements and ensuring scalability. Experimenting with bigrams and sparse matrices aimed to improve model performance, balancing complexity and computational efficiency, and informing our decisions on feature extraction and representation techniques for an accurate and efficient predictor.

As mentioned previously with regards to our simple model, we have set the `random_state=0` parameter in several parts of the code, such as when splitting the dataset and when oversampling the training data. Setting the `random_state` to a fixed value ensures that the results are reproducible. By fixing the `random_state`, we guarantee that the same random seed is used every time the code is executed, leading to identical data splits and sampling.

Furthermore, the dataset is divided into training, validation, and test sets. To balance the classes, the training data undergoes oversampling using `RandomOverSampler`. A `ComplementNB` model is then created with an alpha of 0.1 (determined through hyperparameter tuning detailed at the end of the code). The model is fitted to the oversampled training data.

The model's accuracy is assessed using the validation set, and a classification report is generated. The model achieves approximately 74.83% accuracy on the validation set (please refer to the table for all accuracies achieved). The code also includes a hyper-parameter tuning process involving a grid search over a range of alpha values to identify the best alpha for maximizing the accuracy score.

Naive Bayes classifiers are naturally efficient due to their probabilistic approach and the assumption of conditional independence between features. This allows them to learn from the training data and make predictions fast, without the need for iterative optimization or intricate calculations. In contrast, SVM and neural networks were found to be computationally demanding. The training time for SVM increased significantly with the size of the data-set, making it less suitable for our process. It is worth noting that we had initially made an attempt at utilizing a SVM, however, it did not meet our need for a low runtime, as such we chose not to proceed any beyond an exploratory phase.

As for the choice of Complement Naive Bayes `ComplementNB` classifier, it was specifically selected due to its effectiveness in dealing with imbalanced datasets. In traditional Naive Bayes classifiers, the presence of rare classes may lead to poor estimates of class probabilities. `ComplementNB` is an adaptation of the standard Multinomial Naive

Bayes algorithm that aims to correct this issue. It computes the complement of each class's probability, which helps to reduce the impact of class imbalance and provides a more accurate estimation of rare class probabilities. This characteristic makes `ComplementNB` particularly suitable for our Fake News predictor, where we may encounter imbalanced data with some classes having fewer instances than others. By using `ComplementNB`, we can better model the relationships between features and the target variable.

In summary, our chosen approach offers a straightforward and effective way to create a Fake News predictor, with a focus on balancing classes and tuning hyperparameters to optimize the model's performance. Utilizing bigrams and the `Complement Naive Bayes` classifier provides a solid baseline in comparison to more complex alternatives, while maintaining efficiency and ease of implementation compared to other potential models.

## 4 Evaluation

Firstly, we learned to appreciate the magnitude of data required to train a successful model. As with our first implementation of our pipeline, the runtime severely limited the rate at which we were able to implement changes and receive results and feedback from our code. This experience led us to rethinking our approach to the project with efficiency and optimization in mind, as we would have a tremendously easier time troubleshooting without having to wait large periods of time between iterations.

We found that the performance of both our simple and more advanced models was largely dependant on the input data. This is also argued for in our results on accuracy scores, which can be seen in "Tables" under "Appendix" for visual representation. As we had encountered issues related to the runtime of our pipeline, for reference, our preprocessing framework was only able to generate cleaned data at a rate of 17mb/s, which would have resulted in an overall runtime of the entire pipeline of over a day. As such, it was decided that hence forth further progress was to proceed using a smaller subset of the FakeNewsCorpus dataset. Initially, a method to stream the FakeNewsCorpus dataset into a new csv file was used, where the process would terminate once a desired file size had been reached. However, when the pipeline was executed using this newly generated subset, we found that the data was incredibly unbalanced. This was observed by viewing the confusion matrices generated (for an example of our confusion matrices please refer to the appendix). Upon noting this issue, we hypothesized that randomly shuffling the data would yield a more accurate subset of the dataset, as after manual inspection we found that there was a relatively few instances of reliable articles as opposed to fake ones. After we had run our pipeline using our newly shuffled subset, our accuracies dropped due to the model not being able to blindly predict 'fake' and being correct as fake articles dominated to dataset. Despite this, we observed that the data was being represented far better, and as such our precision and `f1-score`'s improved for detecting reliable articles.

Furthermore, as the shuffled dataset being used is only  $\sim 15\%$  of the entire dataset, we hypothesize that our advanced multi-class classifier model would yield a greater accuracy if given a larger portion of the FakeNewsCorpus dataset. Unfortunately, proving this hypothesis resulted in difficulty due to the aforementioned reasons regarding the runtime when executing the pipeline on the full dataset.

Additionally, throughout the process of creating our fake news predictors as a group of four, we learned to appreciate the vitality of remaining organized. Initially, as a group we had decided to use GitHub (please refer to the appendix for a link) as our primary repository for our project. However, due to unforeseen technical complications, not all group members were able to push and pull from said repository. This resulted in confusion and difficulty managing our files and keeping track of their iterations.

After having executed both our simple and advanced models on the provided LIAR dataset, we were disappointed to see that accuracies our models yielded were subpar. However, upon further inspection of the LIAR dataset, and comparing it to the FakeNewsCorpus dataset used to train and validate our models. We found that the sample size in the LIAR dataset was leagues smaller, resulting in there not being sufficient data for our models to accurately create a verdict. Furthermore, we found that there was a stark difference between the content of the two datasets. Specifically, the content of the FakeNewsCorpus is based on full length articles scraped from the internet, as opposed to the brief statements found in the LIAR dataset. For reference, after manually inspecting the LIAR dataset, we observed that the statements would use as few as three words. Which, as mentioned before, is far less than that our models were trained on.

It is also worth noting that we, as with our initial subset of the FakeNewsCorpus dataset, encountered that the LIAR dataset was also unbalanced. Despite our efforts oversampling the data, the overall accuracy of our models remained below that of when we executed models on the FakeNewsCorpus.

## 5 Conclusion

To conclude, there are a number of reasons why the results on the test set and the LIAR set differed:

**Different data characteristics:** The distributions and properties of the LIAR and FakeNewsCorpus datasets vary, which have an impact on the performance of the models. Due to its emphasis on political fact-checking, as the LIAR dataset is oriented around brief statements that are more complicated and nuanced than those found in the FakeNewsCorpus, this is as much of the information found within the LIAR dataset is implied and not as easily interpretable by our models. For example, please consider as previously stated, that some of the statements contained as few as three words.

**The advanced model might be overfitting the data,** which would explain why it did not perform well on the LIAR Dataset. **Level of difficulty:** While the FakeNewsCorpus contains more blatant examples of fake news, the LIAR Dataset is more difficult for the models to accurately classify because it includes a mixture of true, partially true, and false statements. Furthermore, the labels are mismatched between the two datasets. As the models were trained using the FakeNewsCorpus dataset it performs worse on the LIAR dataset as a result of not being exposed to the unique characteristics of the latter during training if they were trained mainly on that dataset.

Furthermore, the project has given useful insights into the challenges of predicting fake news and the effectiveness of various models on various datasets. The disparity between the performance on various datasets may be unexpected, but it serves as a reminder of how critical it is to comprehend model constraints and the necessity for models to generalize well across various datasets.

Lessons discovered throughout the project:

The performance difference stresses the importance of diverse and representative datasets when training and evaluating models. Data characteristics and quality are also important. The efficacy of a model can be significantly impacted by the quality of the training data.

**Model difficulty:** Better efficiency is not always guaranteed by a more complex model. On various datasets, overfitting can result in bad generalization.

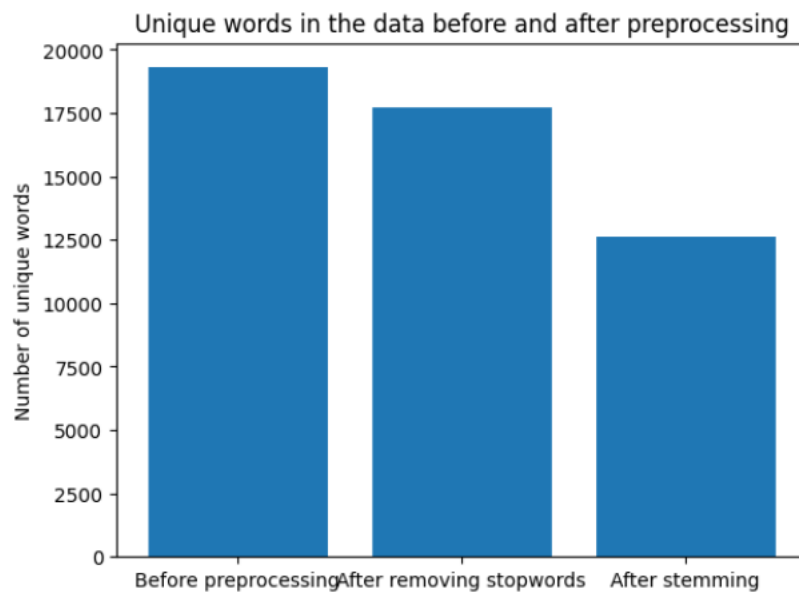
Sometimes a simpler model will do a better job of capturing the fundamental patterns in the data. Better models and statistics will be essential for advancing the field of predicting fake news in the future. The efficacy of the models can be improved by addressing biases, enhancing model architectures, and investigating transfer learning. Curating a diverse, high quality, and representative dataset will be crucial for training more durable models at the same time.

In conclusion, the project has highlighted the significance of model complexity, generalization, and data quality. Better models and better data will work together to advance false news prediction in the future. In order to address the growing problem of false news identification, this project underlines the necessity of ongoing study and innovation in both fields.

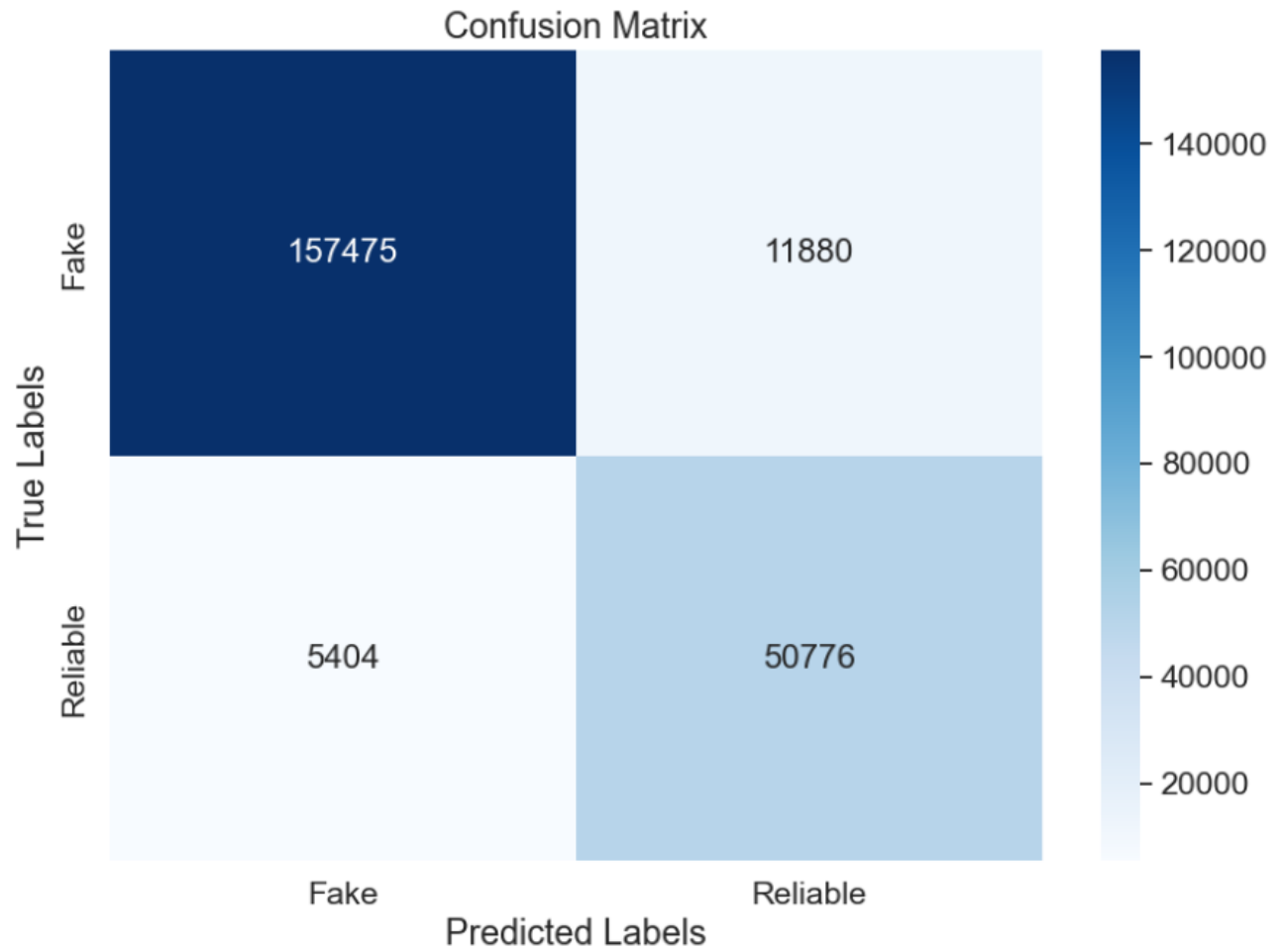
The performance discrepancy between the test set and the LIAR set can be attributed to differences in dataset characteristics, model complexity, and potentially limitations in the training data.

## 6 Appendix

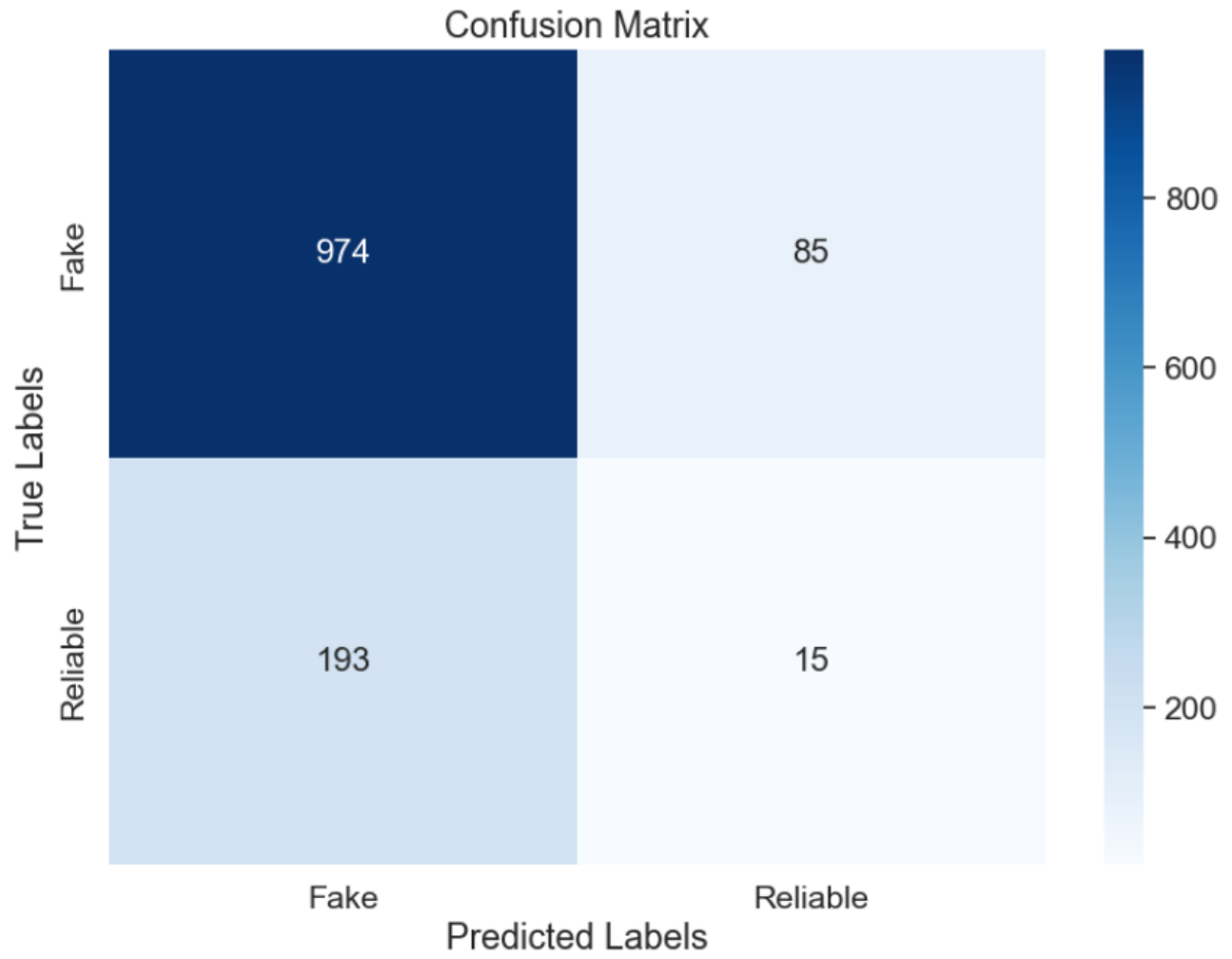
### 6.1 Figures



Part 1, Task 1



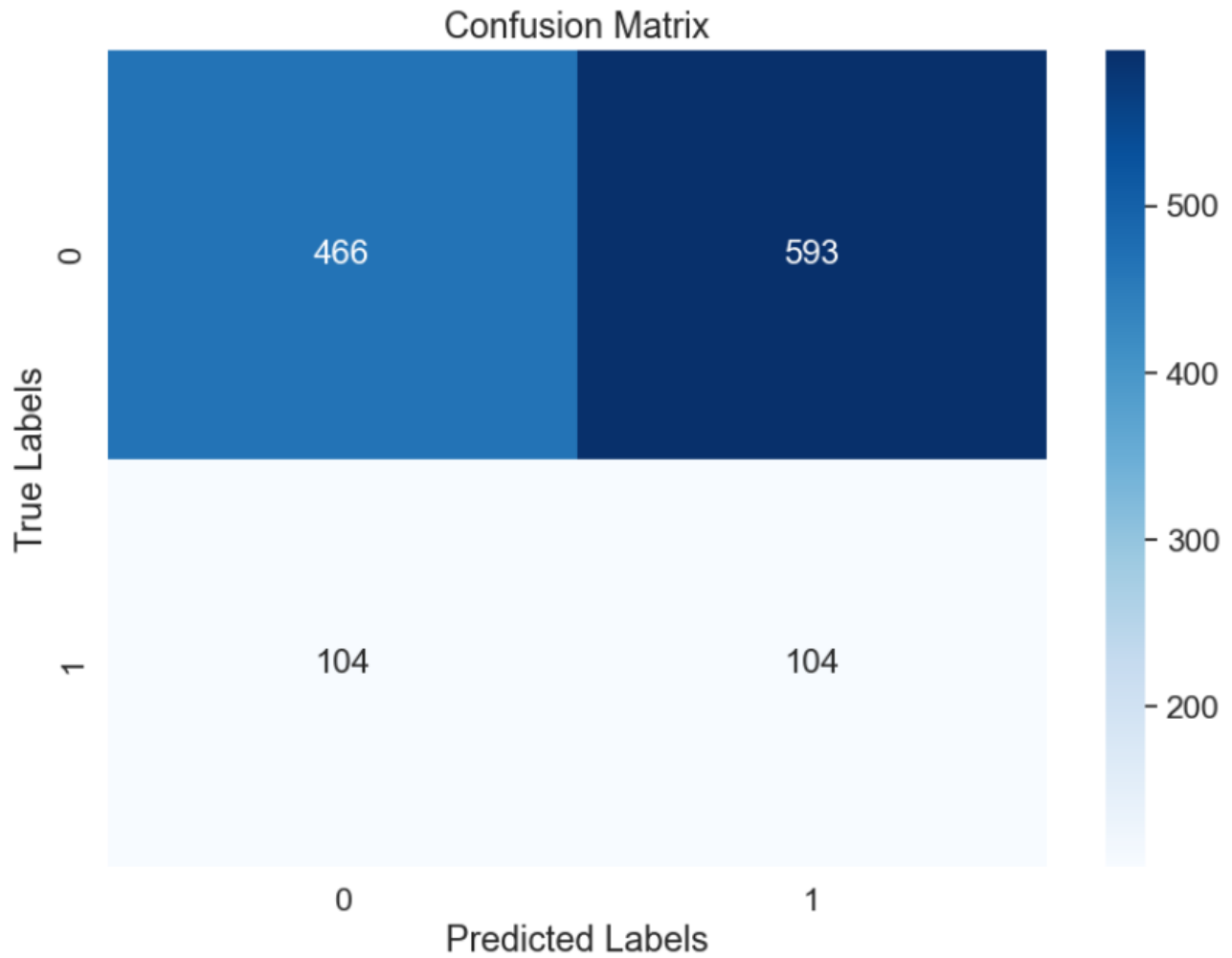




Confusion Matrix

True label	0	1	2	3	4	5	6	7	8	9	10
0	18621	2429	450	120	626	1188	598	177	435	30	1494
1	693	50720	892	94	733	311	491	268	365	25	1588
2	130	863	12565	36	161	106	200	58	68	5	97
3	161	627	295	1499	94	136	75	20	106	7	249
4	1263	3181	563	88	22199	1259	432	480	939	34	3206
5	1358	1553	264	78	936	15596	586	244	308	39	1213
6	169	215	37	19	46	256	2569	18	40	7	116
7	64	423	102	5	102	75	20	1249	36	0	160
8	222	838	159	46	224	188	164	96	4168	21	691
9	164	415	147	66	328	196	56	43	119	6995	287
10	1815	7157	655	180	1744	1586	399	595	1521	44	32753
Predicted label											

Part 4, Confusion Matrix for Advanced Model on FakeNewsCorpus



Part 4, Confusion Matrix for Simple Model on LIAR

## 6.2 Tables

	Simple Model Accuracy Score	Advanced Model Accuracy Score
FakeNewsCorpus	92.3%	74.9%
LIAR Dataset	78.1%	45.0%

## 6.3 Github

<https://github.com/twentytwentycph/dataScienceEksamen>