# FINAL PROJECT: FAKE NEWS PREDICTOR

A PREPRINT

**Peter Bock-Poulsen (sbc605)** **Markus Andersen (dnr319)** **Jakob Karrer (wxp878)**

June 10, 2022

## 1 Know your data

### 1.1 Exploring dataset

We have split our data into six entity sets:

| | |
|---|---|
| **Article** | Core of the data: contains content, type, Id and title. |
| **Author** | Author name and author Id. |
| **Location** | Fields containing articles location, dates for when it was written, scraped and last updated. |
| **Metadata** | Information about content such as summary and length of article. |
| **Tag** | Tag used to mark article. |
| **Meta-keyword** | Keywords and meta-keyword as the difference between them was unclear. |

The ER diagram is shown in figure 1.
We hold first normal form everywhere. We could have broken normal form in author when there are multiple authors to an article, but we gave authors an Id, and created the author article relation. For the most part we uphold the second normal form. However in the location entity class, the Id and the URL together are the primary key. We use the Id to save space in the relation. There is a functional dependency (FD) between the subset of a candidate key, the URL, and the non prime attribute: domain. This violates the Second normal form. We could have fixed this by letting each domain have its own Id in a separate table, and creating a relation between location and domain. The transitive relation `URL` $\rightarrow$ `Domain`, `Domain` $\rightarrow$ `Type`, implies the transitive dependency `URL` $\rightarrow$ `Type`, which exists in our data. However we have split up type and domain in this database meaning we do not model said functional dependency. We believe that the functional dependency `Domain` $\rightarrow$ `Type` would not hold, should new data arrive in the dataset. Therefore the only problem that remains from this transitive relation is what breaks second normal form. We see that the `URL` $\rightarrow$ `Domain` FD is in BCNF however we break it with `Domain` $\rightarrow$ `Type`. Domain is not a Superkey in the location table and therefore we would still break BCNF, even if the other normal forms were satisfied. We see that most of our problems come from the location table and solutions already explained would be sufficient to bringing our schema to BCNF.

In the FakeNewsCorpus we saw that many of the articles had the same content, and we believe that users might have shared the same content through different channels. We have removed these identical articles to ensure that we don't have any overlapping tuples when we are doing our training, validation and testing splits. In addition we discovered, when uploading our data into our schema, that a few of the Ids had words in them. We never found the cause but we believe that it was caused by hidden newlines not caught by `clean-text`. There were only three so we decided to delete them manually.

Our final dataset contains 505,519 unique articles where fake articles make up 55% (279,428 articles) of the total dataset, and real articles make up 45% (227,191 articles).
If we take a look at the amount of words each article contains, we can calculate the skewness using `scipy.stats.skew` and see that the dataset is skewed positively, meaning that a small amount of articles contain a large percentage of the total amount of words in the corpus. This is also apparent when we look at the percentage of the total words the top 20% largest and smallest articles contain1.

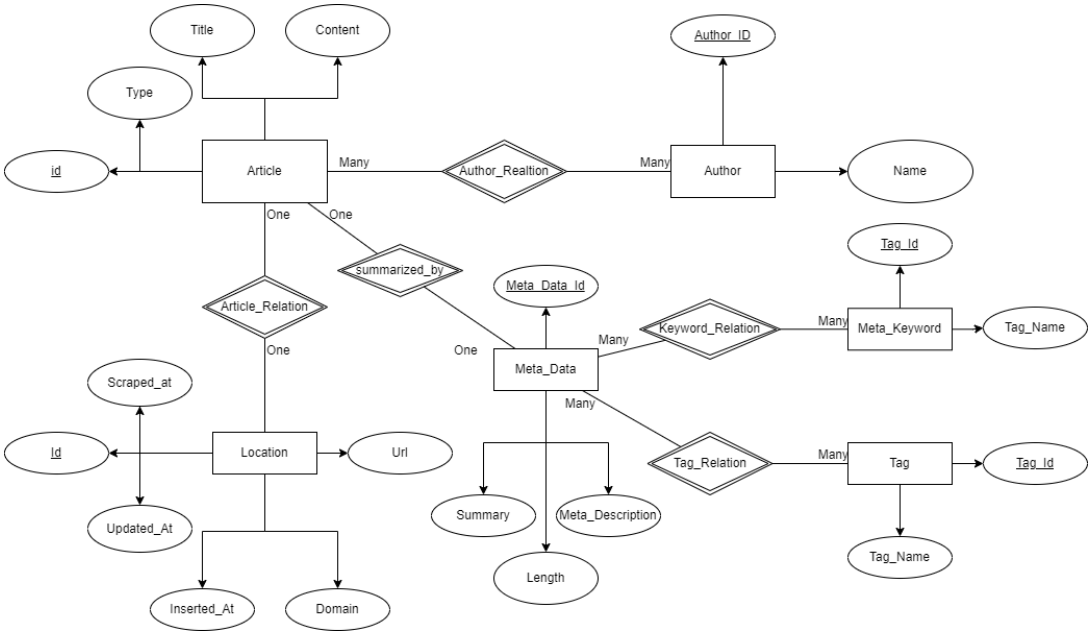


Figure 1: ER diagram

| | **Real articles** | **Fake articles** |
|---|---|---|
| **Skew** | 5.93 | 7.35 |
| **Top 20%** | 54.5% | 54.7% |
| **Bot 20%** | 2.3% | 3.0% |

Table 1: Skewness in FakeNewsCorpus

Using the IQR method[1] with table7 we can calculate that any article containing more than 927 words is an outlier. This means that the 31,331 longest articles are outliers and they count for 28.7% of the total words even though they only make up 6.1% of the articles. This disparity is also apparent in figure 14.

We scraped news articles from wikinews with titles beginning with a letter in the range [A-E] and [T-Z]. The task in itself was solved without larger complications, but it was challenging to navigate through the html code and to find the relevant sections. It took us a while to scrape the 3,608 relevant articles, and whether this was due to our code or just how it works because of the many connections that have to be established, we are not sure. If we had to collect a dataset, the same size as the FakeNewsCorpus, it would take a tremendous amount of time.

### 1.2 Scraped Data Management

We decided that since our scraped articles have both content, type and an author, it would fit within the larger database we used for the FakeNewsCorpus. The location relation as well as the author would be filled out accurately. A lot of the meta data in the FakeNewsCorpus are missing, and hence we do not attach much weight to the metadata. In our baseline models we believe that only the `length` feature is worth including, which we can calculate easily for the scraped data. Since we use the same database for our new data and because task 1.7 is quite ambiguous, we chose to create a view that gives us: content, length and type, from our database. This is all we need for the modelling tasks below.

### 1.3 Foundation for training

We chose content as the feature to train our model on, as it makes the most sense to work with. We could create a complete model by mapping domain names, however for unknown domains this would prove useless. We will only consider the length of the content as metadata, and because of this there is no reason not to include the scraped dataset. We could have created other metrics to add to the metadata class, like the LIX number which might have added predictive power to our model, however we decided to keep it simple, and focus on content and length. Since we have significant hardware and time restraints, we decided to limit our data-set to 50,000 unique articles.

[1] https://cdn.citl.illinois.edu/courses/kines401/lesson1$_{l}ectures/lecture1_{p}5/web_{d}ata/file24.htm$

In appendix A, table6 shows how we mapped the type in the dataset to the binary class. The general idea was to read the documentation from the github repository, to understand the meaning of a label, and then discuss if we think it is closer to real or fake news. We classify political as being real, because it is described as verifiable information. We classify satire as fake, since the description of the label makes it clear that the content includes false information. The label, Rumor, is not described in the documentation and can therefore not be classified, so we don't include it in the dataset.

## 2 Establish a baseline

As baseline models we have chosen Naive Bayes and Logistic Regression. Both models are fast and simple classification algorithms, suitable for high dimensional datasets and with only a limited amount of parameters, which we set to the default values. We see from our data sample that 55% of our articles are labeled fake, and thus we demand our baseline models to have an accuracy > 55%. Otherwise an extremely basic baseline model which predict all articles as fake would be superior.

As text encoding we have applied three methods: TF-IDF, averaging word vectors and Doc2Vec. The accuracy of our baselines are listed in Table 2.

| | **TF-IDF** | **Avg. Word Vec.** | **Doc2Vec** |
|---|---|---|---|
| **Bayes Naive** | 75% | 63% | 56% |
| **Log. Reg.** | 77% | 73% | 70% |

Table 2: Baseline Model Test Results

In the TF-IDF encoding we have limited the number of features to the 20,000 most frequent terms. We have done this to save memory, because our computers couldn't allocate enough RAM for the large matrixes we otherwise would obtain. Many of the omitted features occurs only a few times in our corpus and we hypothesise that it won't have a large impact on our models. This is after taking Zipfs law into account. To further reduce the amount of features we used PCA, but observed that the principal components could only explain a very limited share of the variance. Hence we chose not to use PCA.

We used the gensim library to make our word embedding. We downloaded a word2vec model that had been pre-trained by google on a 100 billon word Google News corpus. We now had our word vectors but how did we get a vector for a document? The most naive way is just to take an average of all the word vectors in an article, which is what we did. This approach does have some weaknesses which we will discuss later on.

The final text encoding was with the doc2vec method[2]. More precisely we used Distributed Memory with Averaging, which acts as memory that remembers what is missing from the current context. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document. We again used the gensim library and trained the model on the corpus 30 times. As Table 2 might suggest the doc2vec model was not trained enough.

As stated previously in this project we chose to include the length of the text in our view, to use on the baseline testing. We thought that the more effort was put into an article the longer it would be, and this could help classify texts. The assumption being that more effort was put into real articles. During our exploration we found that the opposite was actually the case for our corpus. We still use the length but now with a different hypothesis about its implication. We see that Bayes naive reach 79% accuracy and Log. Reg 76%. Slightly better, indicating that in future models it might make sense to include. This however is outside the scope of the project.

## 3 Creating a Fake News predictor

### 3.1 Random Forest

We chose the main predictive models of this project to be a Random Forest (RF), and a Neural network (NN). Both use the implementation provided by `sklearn`. As in the baseline we train, validate and test our model on the three different word encodings using only the 20,000 most frequent features for TF-IDF. We used a training set of size 32,000, validation set of 8,000, and testing set of 10,000. We choose RF because we hoped that the structure of the trees would be able to capture relationships between certain words.

[2]Gensim: topic modelling for humans. (2022). Radimrehurek.Com. https://radimrehurek.com/gensim/models/doc2vec.html

When validating which paramaters we should test our RF model on, we iterated over the following parameters, with all combinations, choosing the ones that yield highest accuracy.

- Split Quality = [Gini, Entropy]
- Max Features = [None, Sqrt, log2]
- Max depth = [10, 15, 20]

We choose the split quality measurement based on the possible inputs of the sklearn function, and the knowledge that deeper trees don't necessarily produce better results[3]. Any other parameters are default. After doing a grid search we find that the optimal parameters are: Split Quality = Entropy, Max Features = Sqrt and Max Depth = 20. The accuracy is displayed in table 3.

### 3.2 Neural Network

For our Neural network implementation our grid search was much more exploratory as we have not gone into great detail, during the course. We found after testing with larger neural networks with hidden layers (800, 400, 200, 20, 2) and (400, 200, 20, 2) that in fact a network with hidden layers (5, 2), performs just as well as the larger ones, and takes less time to train. The alpha parameter did not seem to have any positive impact if made larger indicating that for our data over-fitting is not a problem. We set the max iter parameter to be 6,000 as this yielded the best results. Anything else was set to default, and the testing results are shown in table 3.

| | **TF-IDF** | **Avg. wv** | **Doc2Vec** |
|---|---|---|---|
| **Random Forest** | 80% | 72% | 64% |
| **Neural Network** | 82% | 75% | 66% |

Table 3: Accuracy of Models On FakeNews dataset

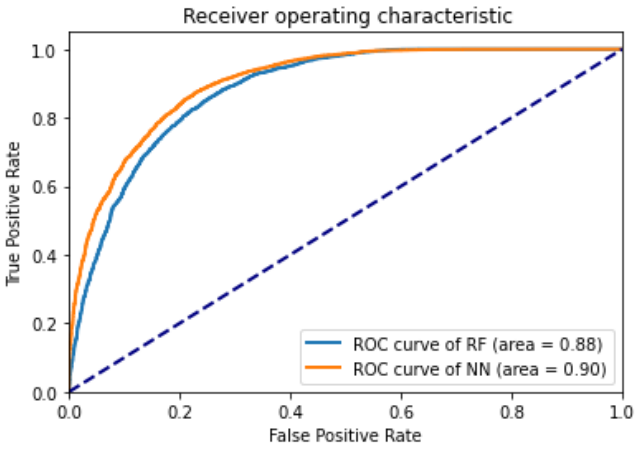


Figure 2: ROC curve of RF and NN (TF-IDF)

### 3.3 Comparing Models

In table 3 we see that in general the TF-IDF encoding performs better for both models. For all encodings we observe that our neural network performs better which is visualized in the ROC diagram in Figure 2. We see that the NN has a higher area under the curve, and thus is better at classifying the real class in the Fake News dataset.
We can't be sure if this is just a coincidence or if that means the model is actually better. Specifically between RF, NN, and the baseline within each encoding: TF-IDF and Avg. Word Vec. If we compare RF and NN it is even more unclear whether this is a significant difference in predictive capability, a result of our sample or chance. We have performed a McNemar's Chi-Square test to test the hypotheses:

$H_0$ : Both classification algorithms have same error rate.
$H_A$ : The two classification algorithms have different error rate.

The McNemars test is suitable since the models are trained and tested on same data and thus dependent. The results of the pairwise tests are listed in table 4.

With a $\alpha = 5\%$ we reject all the null hypotheses and conclude that our advanced models perform significantly better than our baseline models. Furthermore we conclude that the NN performs significantly better than the RF on our test data.

[3]Course: Modelling and Analysis of data - lecture: L10_Classification_Regression_2.pdf, slide 27

| | NN vs. Log. Reg | RF vs. Log. Reg. | RF vs. NN |
|---|---|---|---|
| **Chi$^2$ statistic** | 20.254 | 10.716 | 6.110 |
| **p-value** | 6.78e-6 | 0.001 | 0.013 |

Table 4: McNemars Test Results

So far we have adjusted and evaluated our models purely based on accuracy. We also want to explore how well our model performed in terms of recall and precision. In figure 7 we visualize the binary classifiers, where a real article is considered positive, and fake is negative. Note that the colors scale differently for each graph and comparing them should be done with care. We see in our confusion matrix that for TF-IDF encoding RF actually has better recall of 82%, compared to NN which has 79%. Their F1 score is equal at 79%. We also observe that RF has a precision of 77% while NN has one of 79%. Meaning that 79% of the time we predict that an article is real it is actually real. We could likewise say that 82% of all the real articles RF predicts correctly, which intuitively seems less useful for fake news. Since our dataset is quite balanced and the F1 score of both models are equal, it is hard to say which is better based on these two metrics.

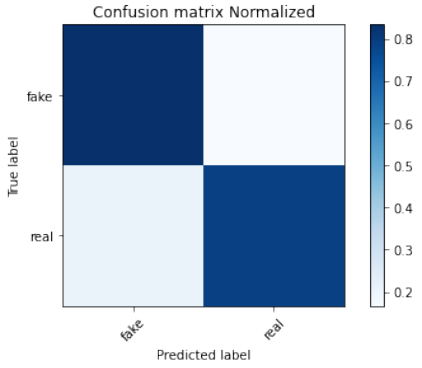


Figure 3: Neural Network TF-IDF

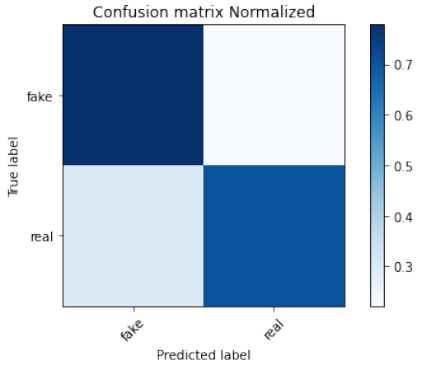


Figure 4: Neural Network WV

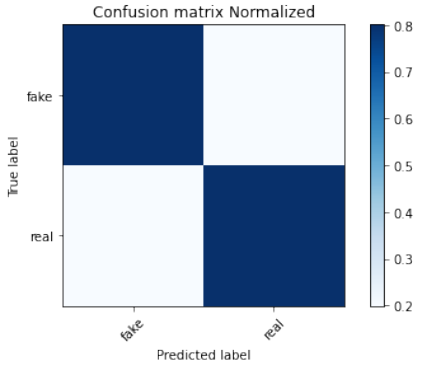


Figure 5: Random Forest TF-IDF

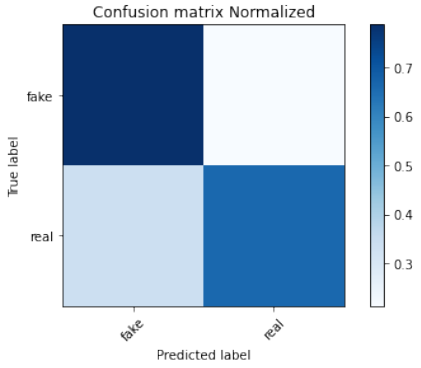


Figure 6: Random Forest WV

Figure 7: Visualisation of Accuracy, Precision and Recall

## 4 Performance beyond the original dataset

We now test our trained models on the LIAR dataset which is completely seperate from our training and testing sets so far, with the hope that our model will generalize well to other data. The handed out LIAR dataset came as three sets: training, validation and testing. We decided that since the first two were not necessary we simply combined them all into one big testing set. After cleaning the data we were ready to test. We chose only to test with TF-IDF and Avg. Word. Vec. encoding since the doc2vec encoding didn't provide any usable results.

We see in table 5 that our models do not work very well on the LIAR dataset. Every model seems randomly scattered around 60% for TF-IDF. The visualisation of precision and recall 13 shows that this comes as a result of our models mostly labeling articles as real. The LIAR set has 71% real articles. Meaning there is something about the

LIAR dataset that makes our models mostly predict real, however sometimes when it is real it predicts fake, seemingly choosing at random. We set; false, barely-true and pants-fire to be fake and the rest real, which could contribute to the distribution.

| | TF-IDF | Avg. vw | FakeNews dataset TF-IDF |
|---|---|---|---|
| **Bayes Naive** | 59% | 44% | 75% |
| **Log. Reg.** | 58% | 57% | 77% |
| **Random Forest** | 57% | 51% | 80% |
| **Neural Network** | 57% | 58% | 82% |

Table 5: Accuracy on LIAR dataset

## 5 Discussion

We observed, when analysing the LIAR dataset, that the length of its content is significantly shorter than the FakeNews-Corpus. This means that our models are trained on data with significantly more information than it is being tested on, as shown in boxplot 8. This means that when we transform the LIAR corpus into TF-IDF we get a matrix with almost exclusively zeros, which would not generalise well to models which are trained with significantly more non zero values. In the boxplot we have excluded the outliers as this would have made it impossible to see the plot of LIAR. See table 7 for the max values.

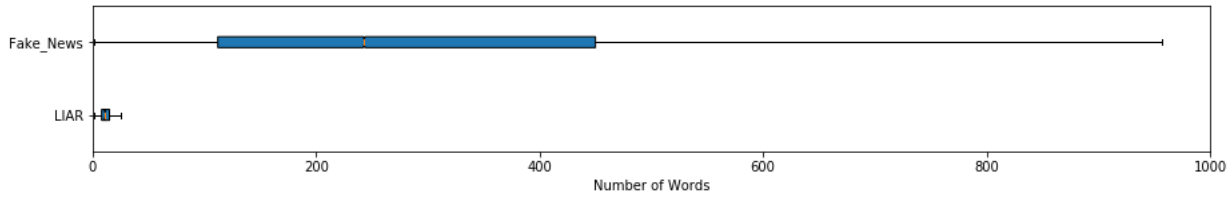


Figure 8: Box plot of content length

When we look back on our project pipeline, we wish we had done some things different. When we cleaned our data, we stored it in the data base and replaced the raw data. Later in the project we discussed that a feature we could add to our model could be the LIX number of an article. A hypothesis we would like to look into is that real articles have a higher LIX number than fake articles. However this is not possible to calculate without punctuation, and a lesson learned is thus always to keep the raw data stored.

In addition we would question our naive encoding with average word vector. We believe that this method loses a lot of the document information. Since we take an average, the order of the words – or the context – is not encapsulated in the final vector. We need the necessary knowledge to find a suitable transformation from a sequence of word vectors to a document representation. That is how to train models with multidimensional features.

Another aspect to criticize is how we limit our vocabulary to the 20,000 most frequent words (after removing stopwords). Perhaps we discard some important words that has importance to either fake or real articles. An approach could thus be to make TF-IDF on differnet chunks of words on the very large vocabulary of the FakeNewsCorpus. The reason we did our dimensional reduction was because some of our models could not take a sparse matrix as input. We could of course make it a requirement, that our models could do this, but we believe this would have been bad practice. From the PCA we saw that the data did not have many, if any, dominating principal components and hence our naive approach was necessary.

We used accuracy as the parameter to decide on the optimal arguments for our models, and to compare them to our baseline models. This however might not be the most useful way to evaluate a model. Recall doesn't seem very useful either as ensuring that we find all the real articles seems unrealistic, and considering the amount of sources on any given subject, unnecessary. Optimising our model for precision would ensure that if an article is classified as real, we know with high probability that we can trust it. In real applications this would be much more useful, since if an article is classified False Negative, we can simply select a new article. If a news article is classified as False Positive however, we decide to trust something that should not have been trusted which is a much more costly mistake. When we were doing the grid search on Random Forest we saw that some combinations of arguments actually yielded upwards of 90% precision. Since we had decided to focus on accuracy, we didn't select this model, as its accuracy was lower. It does however show that high precision might be reasonable goal for later experiments. The choice of performance measurement is, along with the dataset, one of the major issues. What the model is used for in practice, therefore needs to inform which parameters you train for; precision, accuracy, recall or F1-score. The models seem to function well, so we believe that it is the data we train on that cause the issues explained above.

# 6 Appendix A

| Type | Description | Argument | Binary class |
|---|---|---|---|
| Fake | Sources that entirely fabricate information, disseminate deceptive content, or grossly distort actual news reports | A priori fake | fake |
| Satire | Sources that use humor, irony, exaggeration, ridicule, and false information to comment on current events. | Uses false information and exaggeration. Thus the content can not be percieved as real. | fake |
| Extreme Bias | Sources that come from a particular point of view and may rely on propaganda, decontextualized information, and opinions distorted as facts. | Since the information can be decontextualized and opinions distorted as facts, we assume that the content is unreliable in most cases. | fake |
| Conspiracy | Sources that are well-known promoters of kooky conspiracy theories. | We believe that conspiracy is extreme biased, and the argument follows the same as that type. | fake |
| Junk Science | Sources that promote pseudoscience, metaphysics, naturalistic fallacies, and other scientifically dubious claims. | Articles described as scientifically dubious, and hence we believe it must be classified as fake. | fake |
| Hate News | Sources that actively promote racism, misogyny, homophobia, and other forms of discrimination. | The description does not say anything about whether the content is fake or real. We omit to classify this class. | not classified |
| Clickbait | Sources that provide generally credible content, but use exaggerated, misleading, or questionable headlines, social media descriptions, and/or images. | Content is described as credible, and hence classified real. | real |
| Proceed with Caution (unreliable) | Sources that may be reliable but whose contents require further verification. | Since content require further verification, we can not classify this class | not classified |
| Political | Sources that provide generally verifiable information in support of certain points of view or political orientations. | Information is generally verifiable information. | real |
| Credible | Sources that circulate news and information in a manner consistent with traditional and ethical practices in journalism | A priori real. | real |
| Rumor | No description | No information, and thus can not be classified. | not classified |

Table 6: Mapping from Type to Binary Class

# 7 Appendix B

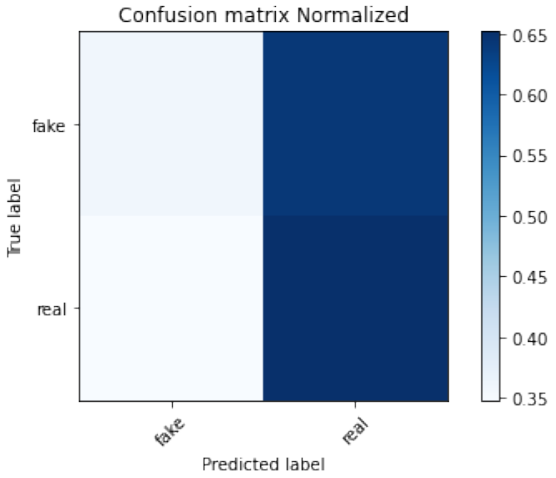


Figure 9: Neural Network TF-IDF LIAR SET

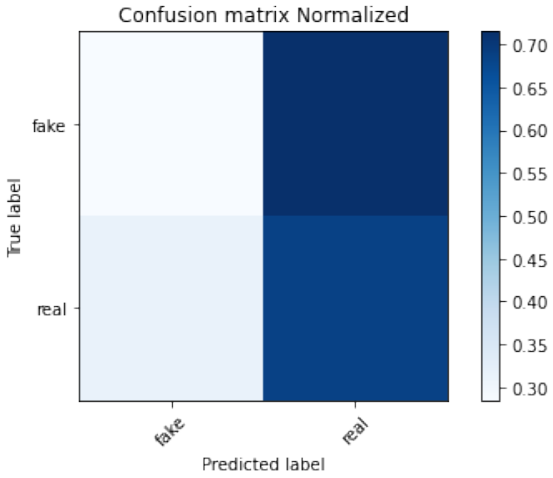


Figure 10: Neural Network WV LIAR SET

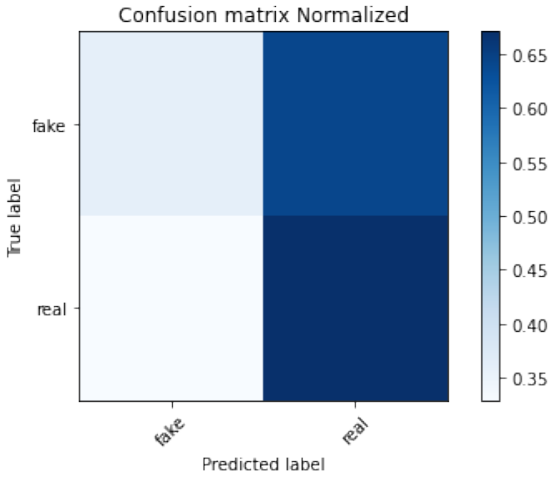


Figure 11: Random Forest TF-IDF LIAR SET

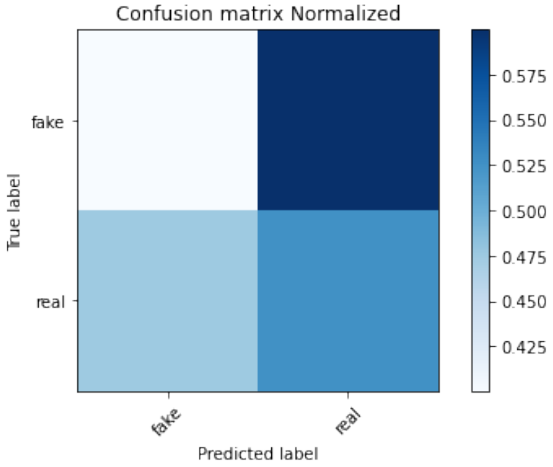


Figure 12: Random Forest WV LIAR SET

Figure 13: Visualisation of Accuracy, Precision and Recall In LIAR

# 8 Appendix C

| | Median | Mean | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|
| Real | 247 | 362 | 1 | 115 | 247 | 440 | 19243 |
| Fake | 239 | 370 | 1 | 111 | 239 | 460 | 17433 |

Table 7: Word Count Distribution

# 9 Appendix D

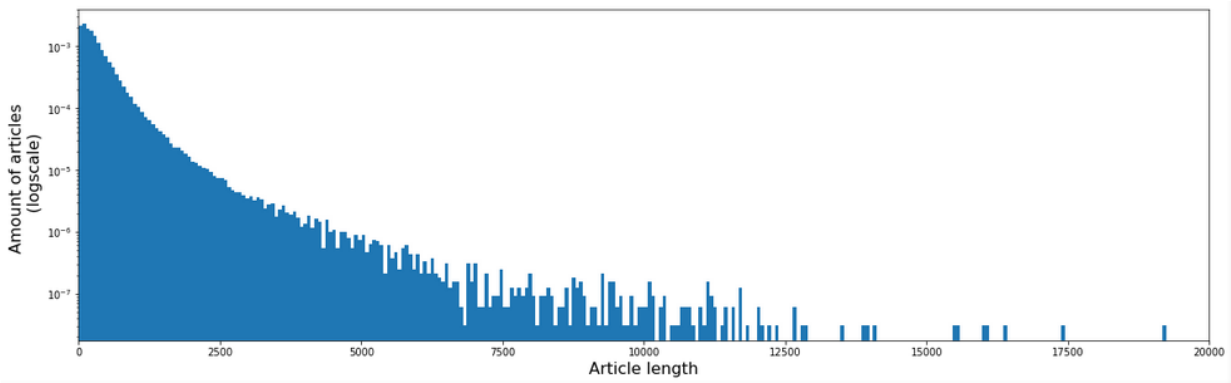


Figure 14: Distibution plot of article length