ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
BUSINESS ANALYTICS Master of Science

Machine Learning and Content Analytics - PT

Too Good to be True

py-truth

Konstantina Georgiopoulou (p2822004)

Anastasios Theodorou (p2822007)

Christos Kallaras (p2822009)

Stavros Kasiaris (p2822022)

September 21

Contents

Abstract

Fake news has become a major phenomenon in our days. In almost every field we observe that there is a lot of misinformation which can be intentional or unintentional. That can have serious consequences in some situations and can lead us to have different opinions about a subject or make wrong choices. As such, we need to build automatic algorithms which can detect fake news at a relatively high percentage. However, in order to have reliable algorithms, we have to be sure that we can exactly define the difference between the meaning of true and fake news. So, the goal is to create a system that take a new sentence or statement and predict if it is true, so can rely on that.

Introduction

World is changing rapidly. No doubt we have a number of advantages of this digital world, but it also has its disadvantages as well. There are different issues in this digital world. One of them is fake news. Someone can easily spread a fake news. Fake news is spread to harm the reputation of a person or an organization. It can be a propaganda against someone that can be a political party or an organization. That was a phenomenon that was observed very often in the 2016 US elections.

The aim of this project is to find models that can predict whether an article is fake or not using Natural Language Processing. For that purpose, a variety of machine learning algorithms are used. The algorithms first have to be trained and validate with data sets called train & validation. After that phase, we use those trained algorithms in the test set in order to evaluate how well they predict.

To reach our final goal, we have to work with text data sets. We encode the categorical features using label encoder, which encodes categorical features as a numeric array with class 0 (fake news) and 1 (true news). Also, we use word and paragraph embeddings. Doc2vec, a paragraph vector, aims at learning how to project a document into a latent d-dimensional space, while word embeddings do the same with doc2vec but in smaller scale (they only take into account the semantics of each word and not of each paragraph).

Detecting fake news is a big challenge because it is not an easy task. The use of Machine learning is proving helpful in this regard. For our project, we use the following algorithms. First of all, we use Logistic Regression, a classifier that is used when the value to be predicted is binary. In our assignment, it can predict or give the result in true or false. Another machine learning technique that we use is Feed Forward Neural Network. This algorithm is biologically inspired classification technique, which is used to learn the relationship between independent variables, which serve as inputs to the network, and dependent variables that are designated as outputs of the network. After that, we model the Recurrent Neural Network (RNN) which is a type of artificial neural network which uses sequential data or time-series data. Finally, we train the BERT model which is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.

In this paper, we will present in detail the way that the data was collected and analysed doing explanatory data analysis. We will discuss the data cleaning phase and after that, we will show the algorithms and all the steps that we follow, and the hyperparameters that we tune. We will present

the results of each algorithm and a discussion about them as well as future work will be made. Finally, we will describe in detail our team members and their roles, and we will close our report with the timeline we followed all the time.

Vision & Goals

In recent years, due to the booming development of online social networks and online media in general, fake news for various commercial and political purposes has been appearing in large numbers and has been spread all over in the online world. An important goal in improving the trustworthiness of information in the online world is to identify fake news timely. Due to this exponential growth of information, our focus on this project is to build a model that predicts fake news. A fake news detection algorithm would be useful in many cases for the online journey of every user. To analyse more, this algorithm would be useful for the social media platforms to detect fake news posts and delete them, in order to prevent the spread of useless information. Further application of the model would be in an aggregator site, in which must be uploaded only trustworthy news.

Inspired by all these, we conclude that an application like the above can be very useful to the world. After searching for areas of study, we understand that has been made a rapid increase in the spread of fake news in the last decade and especially has been observed in the 2016 US elections. For that reason, we are going to search for data sets that are collected for that period in the USA and mostly in the political area. It is also important to mention that such a proliferation of sharing articles online, that does not conform to facts, has led to many problems. These problems do not just be limited only to politics, but cover various other domains such as sports, health, and also science. One such area affected by fake news is the financial markets, where a rumour can have disastrous consequences and may bring the market to a halt.

In our project, we are going to answer some business questions according to our vision. First of all, we want to find which machine learning algorithms can perform well in detecting fake news. How well these algorithms can perform in classify news in a test set and finally which algorithm performs better for our goal.

Methodology

In this part of the assignment, we are going to discuss briefly the methodology that we had used to reach our goals. After we had acquired our data and got acquaint ourselves with them, we had proceeded to the exploratory data analysis to better understand the datasets. Next, we preprocessed the text data, including removing unwanted punctuation, words or symbols that would have an impact on the models. After that, we selected our models (Regression and Neural Networks algorithms). We trained them, after from splitting out data set into training, validation & test using splitting / stratified splitting, in order to keep the same ratio of each label in each dataset. Finally, using many validation techniques, such as the accuracy plots, confusion matrix, classification report & ROC curve we plotted the results of each model, and we proceeded to the model selection process to find the best model for our business goals.

Data Collection

The dataset was acquired by Kaggle. The dataset contains two types of articles fake and real News. This dataset was collected from real-world sources; the truthful articles were obtained by crawling articles from “Reuters.com” (News website). As for the fake news articles, they were collected from different sources. The fake articles were collected from unreliable websites that were flagged by Politifact (a fact-checking organization in the USA) and Wikipedia. The dataset contains different types of articles on different topics, however, the majority of articles focus on political and World news topics.

The dataset consists of two CSV files. The first file named *“True.csv”* contains more than 20.000 articles from *reuters.com*. The second file named *“Fake.csv”* contains more than 17.000 articles. Each article contains the following information: title of article, text, subject type and the date the article was published on.

The data was originally collected by the University of Victoria ISOT Research Lab.

Following are the citations that the creators of the dataset were required:

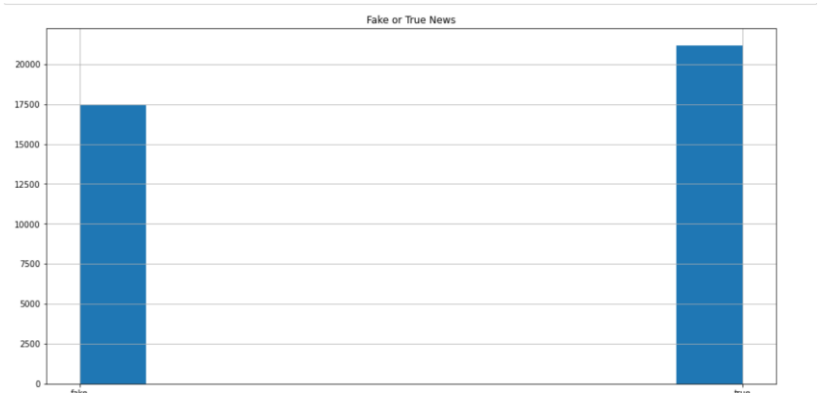
1. *Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.*

2. *Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127- 138).*

Dataset Overview

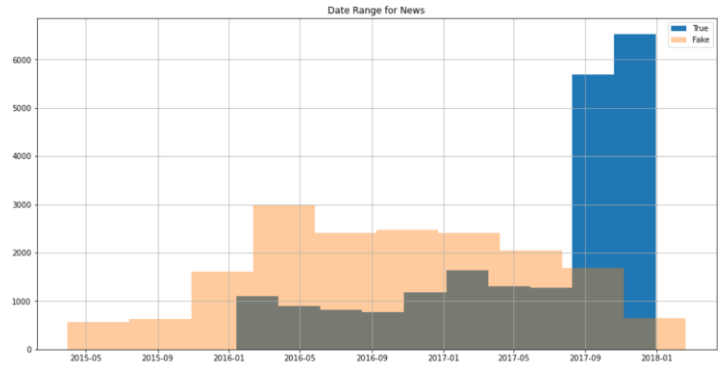
Before we start our analysis, we must explore our data in order to get acquainted with the problem. This pre-processing step is very important because, in exploratory data analysis, we analyze our dataset, summarize the main characteristics, and doing data visualization. This step helps us understand our data and decide the suitable techniques for designing our algorithms.

The first graph is the number of fake and true news.



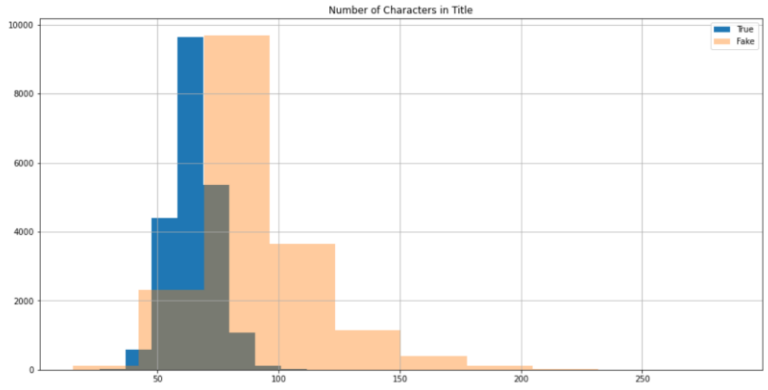
As we can see from the previous graph, the data are balanced. We have nearly the same number of fake and true news, and as a result their number will not have a major impact on our models.

Next, we plot the distribution of fake and true news over the time.



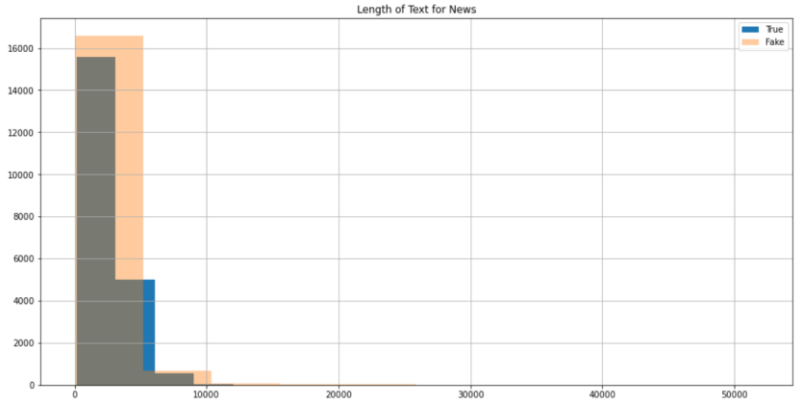
There is the distribution of news across the time frame for Fake news in contrast to True. Given that the "news" in the dataset does not cover the same events, it may give the classification models some trouble, or may be overfit to the dataset.

As we are going to work with text, we want to see the distribution of the number of characters.



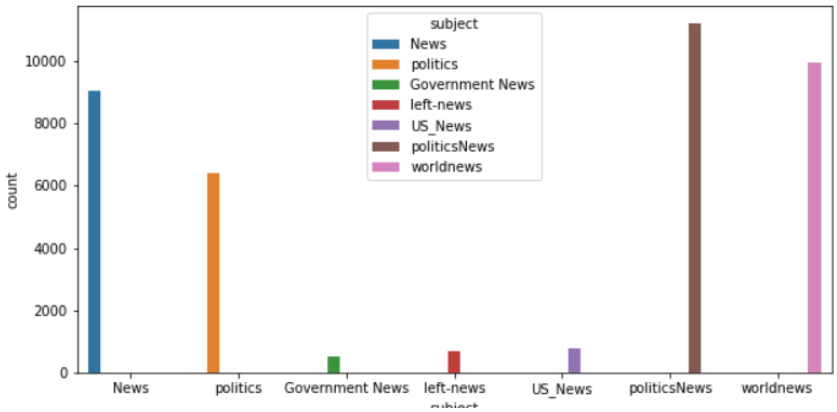
Fake news have a wider range in the length of title than True ones.

We do the same process for the Text.



There are some news that are really long, as we can see from the previous graph.

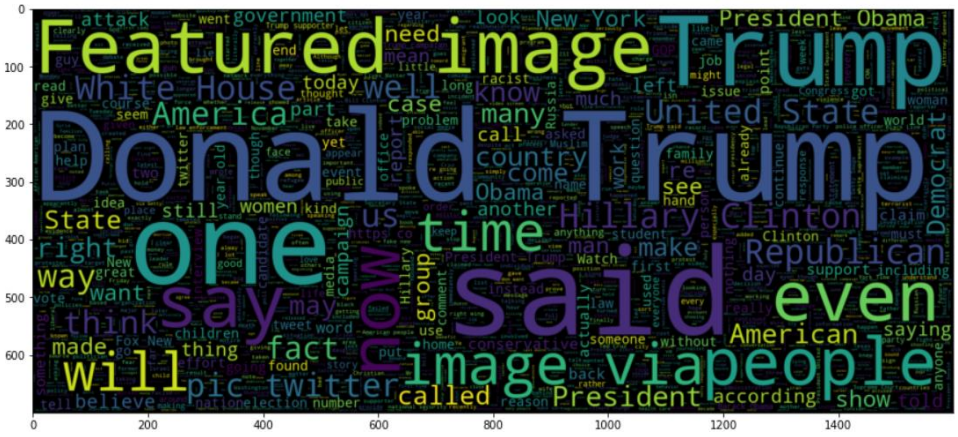
Furthermore, we want to plot the distribution of the news across the different subjects.



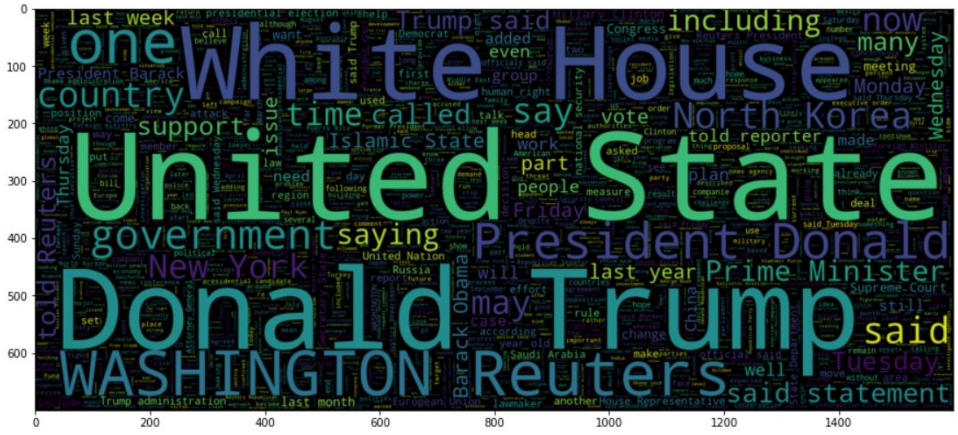
On the previous plots, we can conclude that we have mainly “politicsNews” and “worldnews” rather than “Government News” and “Us News”.

Last but not least, we do the worldclouds that depict the words that appear most on the fake and on the true news.

- WorldCloud for Fake News:



- WorldCloud for True News:



Data Processing

After we have explored our data, we have to do the data cleaning. This process is very crucial for our analysis because we fix or remove incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within our dataset.

1. **Links**

First, we have to see if there are URLs in the text and title columns. Using the regular expression command, we located links, probably from profile photos. We think it would be more significant if there were links in a more compact way , so we will convert all the urls to the placeholder {link}. It is also important to mention that there were no links in the title.

2. **Twitter Users**

As we know, most of our data are connecting with Twitter. That means that in our text we have mentioned Twitter Users. Using the regular expression command, we can understand that indeed there are many mentions in the text column, especially for @realDonaldTrump, who was the President of the United States, during most of the timeframe of the dataset. The way we handle this is by replacing all the Twitter handles with @twitter-handle.

3. **Capitalization**

Because words with all caps are an import way that emphasis is made online, we want to keep words that are in all caps while making all the letters in other words lower case. Words of length of one will be made lower case though since they are likely *A or I* which can be made lowercase without losing much emphasis.

4. **Numbers**

Numbers do not seem likely to indicate Fake news, although certain dates or numbers may. The only date/number we've come across that may have significant meaning is 9/11. We will change it to *nine-eleven* so that numbers can more easily be removed.
We will replace the numbers with a space because some of the sentences run together and end with a number. Replacing the number with a space will split the sentences.

5. **(Reuters)**

Almost all the True news stories have (Reuters) at their beginning and a ML model would merely learn that as how to distinguish Fake vs True, which would overfit potential models to this dataset. So, we remove "(reuters)" from the news.

6. **Punctuation and Single Letter Tokens**

We remove all of the Punctuation tokens except for the exclamation point, because it seems like it may be an indicator of Fake news. We also remove all the single characters except for *i*.

7. **Remove "*'s*"**

While the fake news frequently or always didn't remove the apostrophe from *"'s"*, it doesn't look like that was done to the true news. *"'s"* will need to be removed so that it doesn't become a false indicator of true news.

8. **Remove Date Words**

To better generalize the models that we are going to use, we remove all date words.

After we have done all these data cleaning steps, we save our new data frame for the use of our algorithms in the next steps. It is worth mentioning that normalization wasn't necessary in this dataset, because all the news have more or less the same form.

Algorithms, NLP architectures

Word Embeddings

Since the dataset consists of text (news article written in the English language), before we train and fit our models, we need to transform the data into a form that can be understand and edited by the models. An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings are very useful for representing words, because they take into account the semantics of each word. Embeddings turn text into sequence of numbers, in other words categorical data into vectors.

Document Embeddings – Doc2Vec

Another approach is Doc2Vec, an NLP tool for representing documents as a vector. In general, since models cannot be used with words we need to label encoding them. However, when using such encoding, the words lose their meaning. For example, if we encode Greece as id with value 1, Athens with id 3 and power as id 5, Greece will have the same relation to power as with Athens. We would prefer a representation in which Greece and Athens will be closer than Greece and power. So, we need a numeric representation for each word, that will be able to capture such relations as above. For that purpose, "Word2vec" was presented in 2013. "Doc2vec" is a generalization of "Word2vec" for documents. The goal of "Doc2vec" is to create a numeric representation of a document, regardless of its length. But, unlike words, documents do not come in logical structures, that is why a new method was created. There are 2 methods for doc2vec:

A. Distributed Bag of Words (DBOW): The paragraph vectors are obtained by training a neural network on the task of predicting a probability distribution of words in a paragraph given a randomly sampled word from the paragraph.
B. Distributed Memory (DM): DM acts as a 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.

Logistic Regression

Our first approach was a Logistic Regression model. Logistic regression is a process of modelling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome: something that can take two values such as true/false, yes/no. In our case the binary outcome was true or false depending on the news article.

For the data to be proper tokenized we used Doc2Vec as mentioned above. For the first try we used doc2vec with DBOW as method with 300-dimensional feature vectors. As for workers we use the system core threads to train the model. We then trained the model with the training tagged dataset for 30 epochs and stored the results in a vector. We used that vector to fit the Logistic Regression model. The accuracy was at 60%. Next, we did the same but with DM as method for our doc2vec model. The results were better this time (91% accuracy). For our final try we created a doc2vec

model that combines the previous 2 models and ultimately had the higher accuracy (92%). The detailed evaluation of the models will be presented later (section *Results and Quantitative Analysis*).

Feed Forward Neural Network (NN)

After that, we used Feed Forward Neural Network to train our data and predict the class for every news. This approach is one of the simplest in artificial neural network, because the information moves only in one direction, from one layer to another layer. The information goes only forward, which means that every layer takes as input the output of its previous layer.

In our case, after splitting the data in stratified way (60% training set, 25% test set and 15% validation set) we used two methods for processing our data before building our model.

- ***Label Encoder***

This method creates a vector for every record of the dataset and stores these records in one big sparse matrix being labelled based on the class of each record (0 for fake news and 1 for true news). This matrix consists of columns equal to the number of different words that our dataset contains and rows equal to the number of our news. So, in the end this matrix had size of 24631 rows and 15000 columns. We had only 15k different words, because we have set as vocabulary only the first 15k most common words. In other case, if we used all the different words of our dataset, then the algorithm maybe will have needed a lot of hours or even days in order to be completed.

Later, we built our sequential model with **3** dense layers, **2** activation functions (one "*Relu*" and one "*Sigmoid*") and **2** Drop-out layers. In this part it is worth clarifying some terms:

- **Sequential model:** in this structure every output of each layer is the input of the next layer
- **Dense Layer:** is a deeply connected neural network
- **Activation Function:** functions that decide whether a neuron should be activated or not. This is happening depending on the result of each neuron derived from a function (this function takes into account inputs and weights).
    - **Relu:** deactivates those neurons are less than 0
    - **Sigmoid:** depending of the output of the function gives 0 for small values (usually negative) and 1 for bigger ones (or positive).
- **Drop-Out Layer:** in every step sets the 40% of the neurons as 0 in order to prevent overfitting due to the fact that some nodes gather after some repetitions too much information.

The above model was trained **30** times (**epochs**).

Later, we compiled the model using Binary cross entropy loss function, because we wanted to predict only 2 classes ('fake' or 'true' news) and Adam optimizer, an optimizer more robust than others which uses the gradient descent and goes steadily to global minimum loss avoiding trapping in local minimums.

- ***Word Embeddings***

As we mentioned before, this approach converts categorical data, such as text or more specifically sentences into vectors. So, we used this technique to convert the words of our text into tokens and then feed them into our feed forward neural network. We started setting as maximum words that

our model will train the first most common 15 thousand words. We selected this number, because we thought that gives us a good representative of our dataset and the results in the evaluation (is shown in the next paragraph) are pretty satisfying. Despite of that, we also selected as maximum length of our sentences the mean words of all the sentences contained in the dataset, which is 401 words. In other words, the algorithm cuts those sentences that are too big (above 401 words) and for those that are too small adds zeros. This act was necessary in order to continue with our model, because we want each sentence in our tokens to have the same length.

Finally, we created our model which contains **1** embedding layer, **1** flatten layer, which converts the data into a 1-dimensional array for inputting it to the next layer, **1** dense layer with "Relu" as activation function, **1** Drop-out layer with 40% of neurons being setting off and **1** last dense layer as output with Sigmoid as activation function. We used again 30 epochs to train the model and for the compiling we used the same parameters as the ones in the label encoding (Binary cross entropy loss function and Adam optimizer).

Recurrent NN

A recurrent neural network (RNN) processes sequences of one element at a time while retaining a memory (called a state) of what has come previously in the sequence. Recurrent means the output at the current time step becomes the input to the next time step. At each element of the sequence, the model considers not just the current input, but what it remembers about the preceding elements. This memory allows the network to learn long-term dependencies in a sequence which means it can take the entire context into account when making a prediction, whether that be the next word in a sentence or a sentiment classification. The general idea is that RNN's are designed to mimic the human way of processing sequences.

At the interior of an RNN is a layer made of memory cells. One popular and efficient cell is the Long Short-Term Memory (LSTM) which maintains a cell state as well as a carry for ensuring that the signal (information in the form of a gradient) is not lost as the sequence is processed. At each time step the LSTM considers the current word, the carry, and the cell state. The LSTM has 3 different gates and weight vectors: there is a "forget" gate for discarding irrelevant information; an "input" gate for handling the current input, and an "output" gate for producing predictions at each time step The function of each cell element is ultimately decided by the parameters (weights) which are learned during training.

We are using the Keras Sequential API which means we build the network up one layer at a time. So, we start with the Embedding layer which maps each input word to a 300-dimentional vector and it uses pre-trained weights which we supply in the "weights" parameter. We also set trainable to "False" in order not to update the embeddings. After that, we set a layer of LSTM cells and since we are using only one LSTM layer, it does not return the sequences. Next, we use a fully connected Dense layer with "relu" activation which adds additional representational capacity to the network. By using a Dropout layer, we prevent overfitting to the training data and then we also use one more Dense layer with "sigmoid" activation. The model is compiled with the Adam optimizer and trained using the binary cross entropy loss.

Before training our model we used early stopping , which halts training when validation loss is no longer decreasing. The batch size is 64 and the model was trained 30 times.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is an open-source machine learning framework for natural language processing (NLP). It makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all its surroundings. For this model we used the pre trained based BERT model "bert-base-uncased" on English language using a masked language modelling (MLM) objective. This model is uncased: it does not make a difference between english and English. The model was extremely slow when using the whole cleaned text from the dataset. Specifically, it did not finish even after 10 hours of running so we used only the cleaned title in which the model finished after approximately 5 hours.

Results & Quantitative Analysis

In this paragraph we are going to describe the evaluation of the models and we are going to analyse how good they fit to our dataset. First, we will present the evaluation metrics and later we will compare them, in order to end up with the best model so far.

Logistic Regression

- **Doc2Vec with DBOW method**

Confusion Matrix and Accuracy

As mentioned earlier the Accuracy for the Logistic classifier using a DBOW doc2vec model was at 60% meaning that 60% of the predictions were correct. For a better understanding of the results we can take a look at the confusion matrix that has the totals of predicted and actual values:

	fake	**true**
fake	1197	1420
true	870	2309

The model accuratly predicted 1197 fake news from total 2617 and 2309 real news from 3179.

Classification Report

In the next figure we can see the classification report. Before analysing the classification report it is worth mentioning the definitions of the below metrics:

- **Precision**: the number of correct positive results divided by the number of positive results predicted by the classifier.
- **Recall**: the number of correct positive results divided by the number of all samples that should have been identified as positive

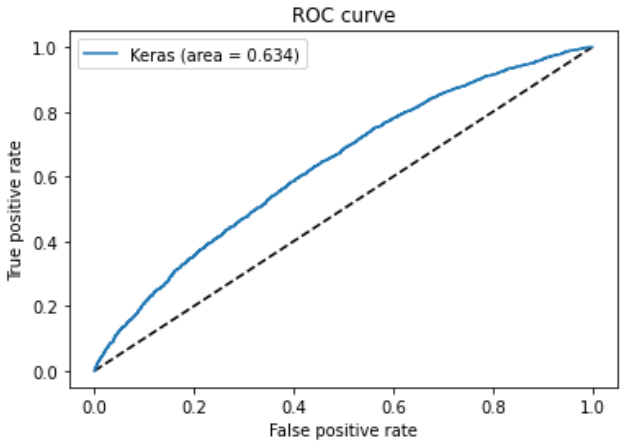
- **F1 Score**: the Harmonic Mean between precision and recall. It shows how precise our classifier is (how many instances are classified correctly), as well as how robust it is (it does not miss a significant number of instances).
- **Support**: the number of news.

	precision	recall	f1-score	support
fake	0.58	0.46	0.51	2617
true	0.62	0.73	0.67	3179
accuracy			0.60	5796
macro avg	0.60	0.59	0.59	5796
weighted avg	0.60	0.60	0.60	5796

From the above report we can conclude that identification of 4 of every 10 fake news is incorrect, and 6 is correct. This means that our model is a medium one and we cannot be precise and confident for the final outcome of an unknown article.

ROC Curve

The final evaluation metric for our model is the **ROC** curve. The ROC curve shows the trade-off between sensitivity and specificity. Area Under Curve (**AUC**) of a classifier is the probability that a classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. So, models that give curves closer to the top-left corner indicate a better performance. In other words, their ratio true positive to false positive should have the biggest possible value.



The curve is close to the diagonal (no predicted value) so the model does not seem to have a good performance.

- **Doc2Vec with DM method**

Confusion Matrix and Accuracy

After training this model we can see that the accuracy is 91% and from the below confusion matrix, 2336 fake news and 2961 real news were predicted correctly.

	fake	true
fake	2336	281
true	218	2961

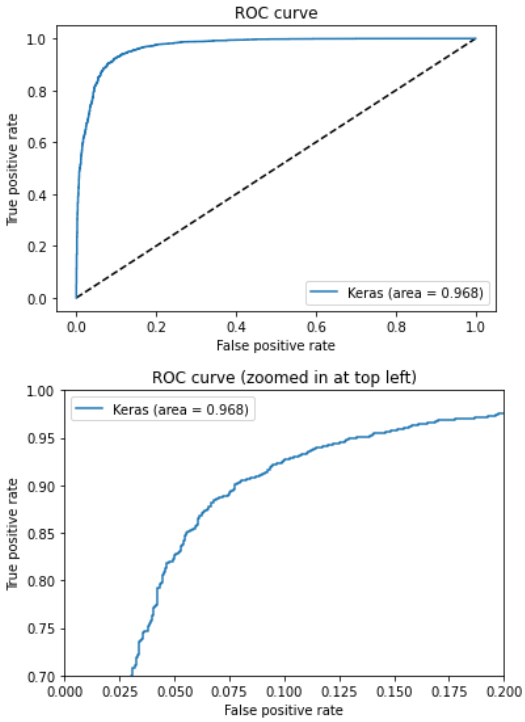
Classification Report

The results were better when using Doc2Vec model with DBOW method (almost all the scores have values near 90%).

	precision	recall	f1-score	support
fake	0.91	0.89	0.90	2617
true	0.91	0.93	0.92	3179
accuracy			0.91	5796
macro avg	0.91	0.91	0.91	5796
weighted avg	0.91	0.91	0.91	5796

ROC Curve

From the next figure we can see that the ROC curve is also improved since it is closer to the top left corner.



Overall, this model seems to have a good performance.

- **Doc2Vec with DM and DBOW methods.**

  Confusion Matrix

The following confusion matrix indicates that 2355 fake news and 3000 real news were predicted correctly.

	fake	true
fake	2355	262
true	179	3000

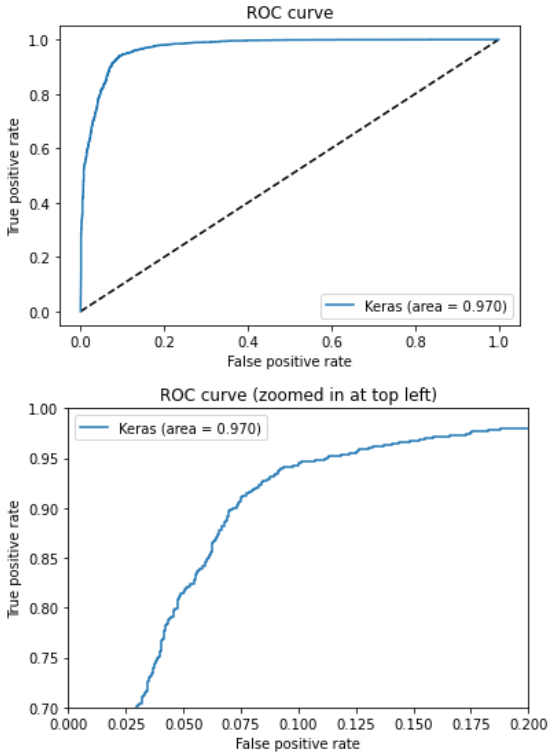
Classification Report and Accuracy

The results were slightly better using a combination of DM and DBOW methods. Accuracy was at 92% as can be seen in the next classification report.

	precision	recall	f1-score	support
fake	0.93	0.90	0.91	2617
true	0.92	0.94	0.93	3179
accuracy			0.92	5796
macro avg	0.92	0.92	0.92	5796
weighted avg	0.92	0.92	0.92	5796

ROC Curve

Finally, the ROC curve is closer to the top left corner than the previous 2 models, indicating good performance (ROC Curve Score: 97%)

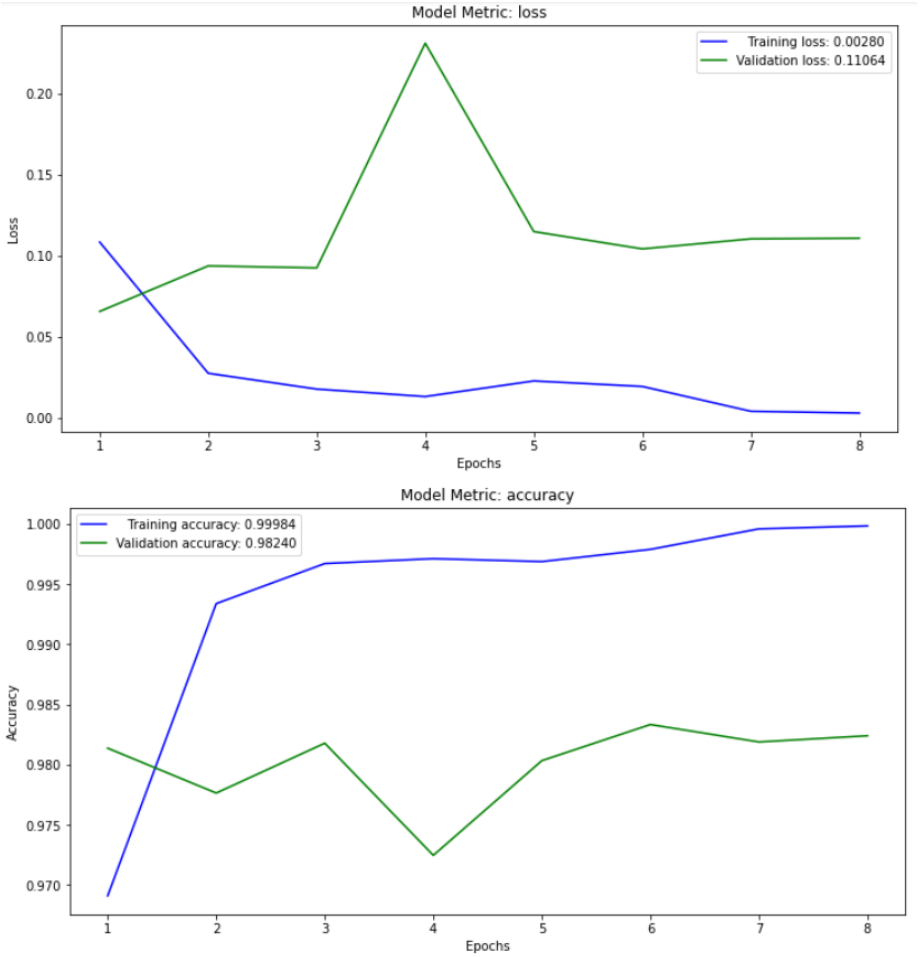


In summary, using Logistic Regression with a Doc2Vec model we can conclude that the worst scores has been found with DBOW method, while using the combination of DM and DBOW models we had the best scores (92% accuracy).

Feed Forward NN

- ***Label Encoder***

  Binary Cross Entropy Loss Function (a function that shows the error) and **Accuracy** for training and test sets after every epoch.



  Binary Cross Entropy Loss and **Accuracy** for validation set

```
Test binary_crossentropy: 0.0731

Test accuracy: 98.137 %
```

First of all, the loss function, takes very small values. This is more than good, because it shows that the error for predicting the right class in our binary classification problem is minor. Combining with the fact of both the training and the validation loss have very low values (from about 0,11 to 0,07) it is a good indicator of a very promising model. We have to clarify that the training loss indicates how well the model is fitting the training data, while the validation loss indicates how well the model fits new data.

On the other hand, the accuracy metric is really high (above 98% for both training and validation). This means that our model is very accurate and predicts correctly the class of the news (fake or true) nearly all the time.

Confusion Matrix

	fake	true
fake	1939	24
true	57	2327

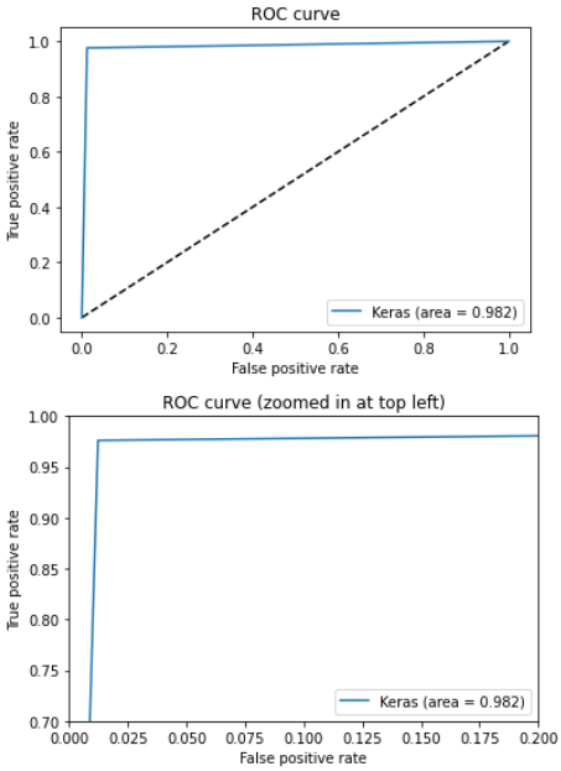
From the above confusion matrix, we can observe that the performance of the model is pretty good. This is concluded by the fact that the predicted values do not differ a lot from the actual values. For example, from the 1963 false news, only the 24 predicted falsefully as true and from the 2384 true news only 57 predicted as fake.

Classification Report

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1963
1	0.99	0.98	0.98	2384
accuracy			0.98	4347
macro avg	0.98	0.98	0.98	4347
weighted avg	0.98	0.98	0.98	4347

After observing the results of our classification report we can assume that our model performs really good (all records are from 97 to 99%).

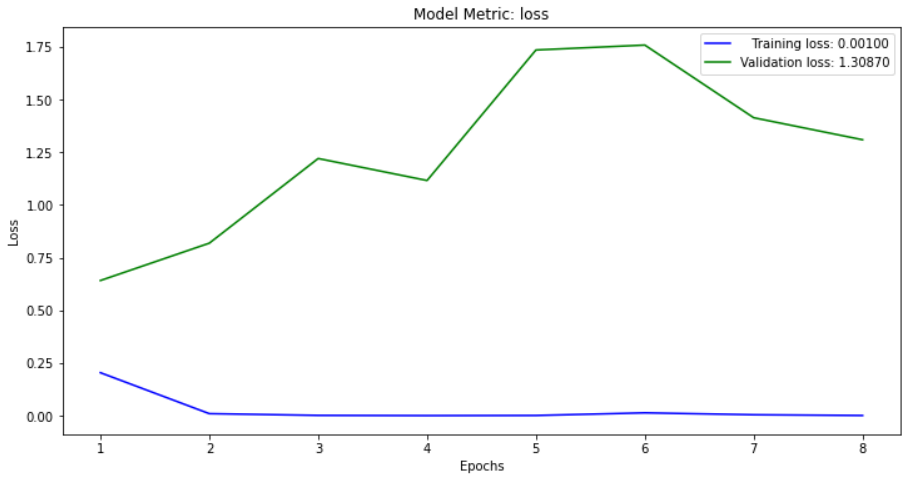
ROC Curve

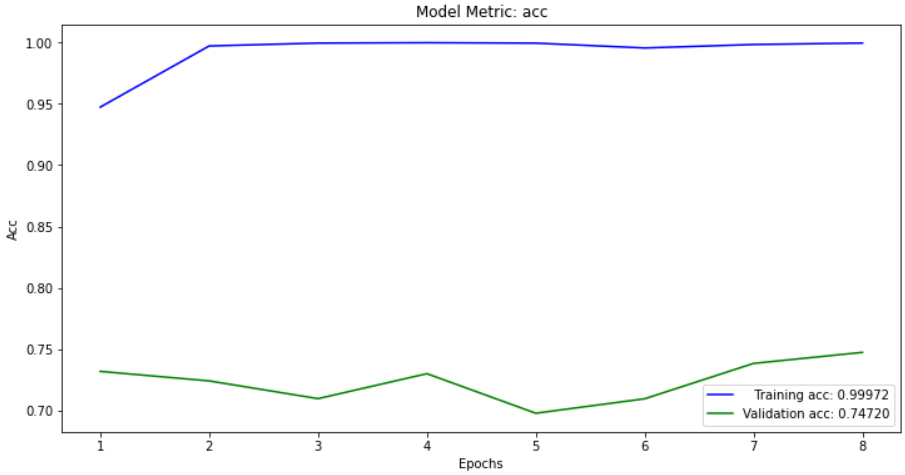


ROC Curve and its score (98,19 %) indicate nearly perfect performance of our model.

- **Word Embeddings**

  Binary Cross Entropy Loss Function and **Accuracy** for training and test sets after every epoch





Binary Cross Entropy Loss and **Accuracy** for validation set

```
Test binary_crossentropy: 0.7003

Test accuracy: 70.393 %
```

In this case we can observe that the model was created with embeddings is worse than the previous one as far as the loss and the accuracy in the training and test set. We can see 1,31 loss on validation set, 0,70 loss on test set, approximately 75 % accuracy on validation set and only 70,4% accuracy on test set. These differences are pretty big comparing with the previous model, which had very satisfying results.

Confusion Matrix

The below confusion matrix depicts the medium evaluation metrics found above. We can observe that from the 1963 false news the majority of them (1031) predicted falsefully as true and from the 2384 true news the 256 predicted as fake.

	fake	true
fake	932	1031
true	256	2128

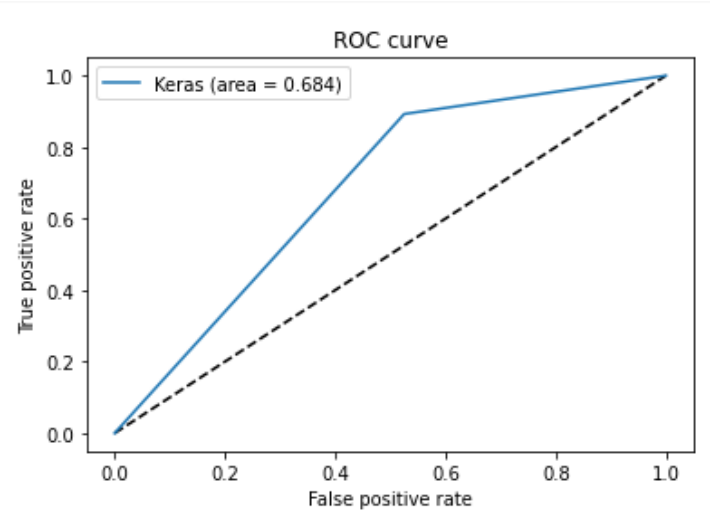
Classification Report

	precision	recall	f1-score	support
0	0.78	0.47	0.59	1963
1	0.67	0.89	0.77	2384
accuracy			0.70	4347
macro avg	0.73	0.68	0.68	4347
weighted avg	0.72	0.70	0.69	4347

We can take a closer look on evaluation metrics from the above classification report. In it we can see that for the 0 class (fake news) the precision, recall and f1-score are much lower than in the one model found previously (78%, 47% and 59% respectively). The same more or less is also observed in the true news where in this case we have 67% precision, 89% recall and 77% f1-score.

ROC Curve

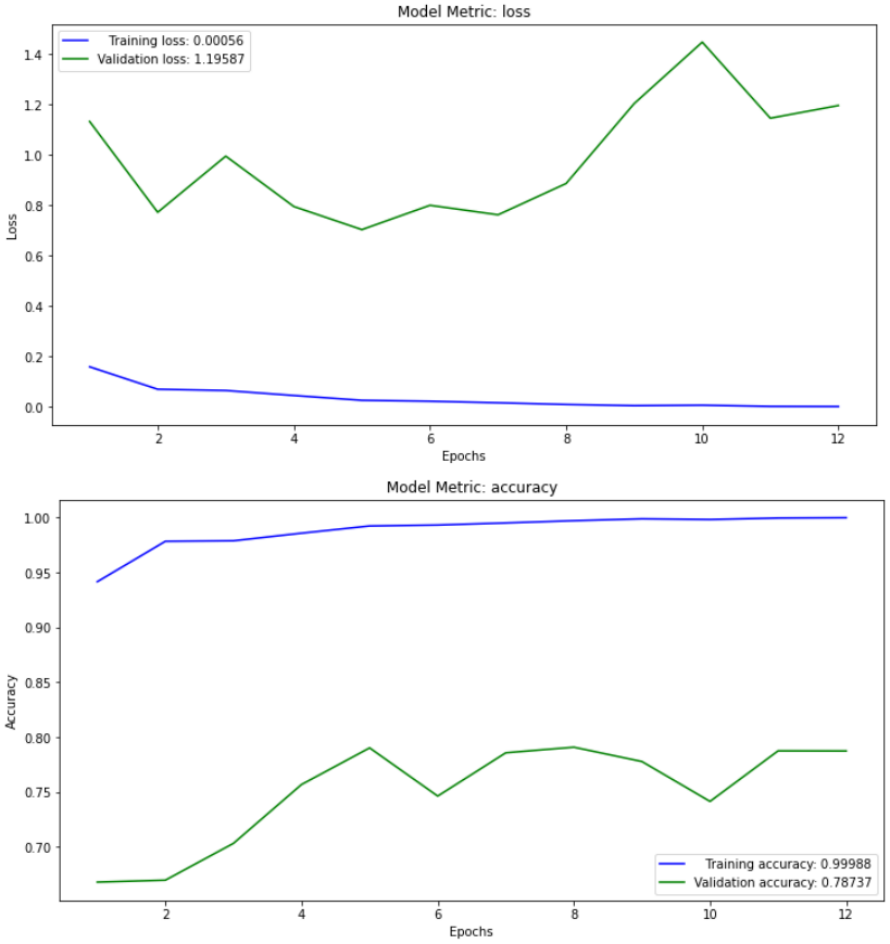
As we can see below the ROC Curve has score 68,37 %, worse than the one found with label encoding.



Concluding, from the two different trained ways in the Feed Forward NN, the word embeddings is less effective and gives less accurate and precise results. It is worth to mention in this part, that generally label encoding as way of converting data to be inputted in NN, is not so effective as the embeddings, because it needs a lot of memory and does not take into account the semantics of each word. But, in our case we can see that this is not happening and with that way our model predicts nearly all the news in the right category. One possible explanation for this, is that with the embeddings some words with the same meaning have been gathered close together, something which is misleading in our case and confuses the algorithm a lot. For example, a word that have been found in the true news may also have been found in the fake news as well, a fact where in the embedding way gives us low evaluation metrics.

Recurrent NN

Binary Cross Entropy Loss Function and **Accuracy** for training and test sets after every epoch



Binary Cross Entropy Loss and **Accuracy** for validation set

```
Test binary_crossentropy: 1.2156

Test accuracy: 65.332 %
```

From both two metrics in all the sets trained we can conclude that the loss takes 1,2 value on validation and test sets, something which is bigger than all the previous models. This fact with combination with the medium accuracy (79% on validation set and 65% on test) shows that the model does not classify correctly the majority of the news.

Confusion Matrix

	fake	true
fake	1487	476
true	1031	1353

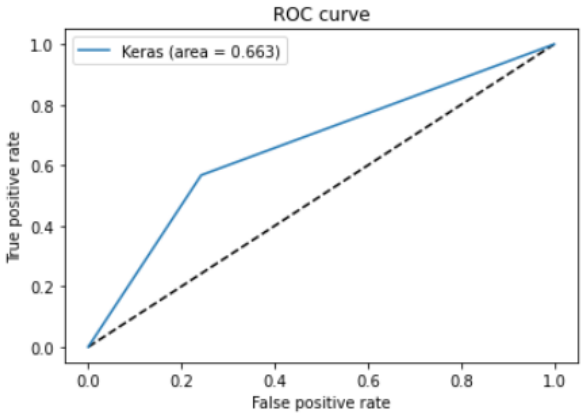
This confusion matrix shows that from the 1963 false news the 476 predicted falsefully as true and from the 2384 true news about half of them (1353) predicted as fake.

Classification Report

	precision	recall	f1-score	support
0	0.59	0.76	0.66	1963
1	0.74	0.57	0.64	2384
accuracy			0.65	4347
macro avg	0.67	0.66	0.65	4347
weighted avg	0.67	0.65	0.65	4347

Mediocre values can also be observed in the above classification report (from 57% to 76%).

ROC Curve



Finally, the ROC curve depicts the bad fitness of the model to our dataset. Its value of 66,25 % shows us a medium model.

BERT Model

Confusion Matrix

The following confusion matrix indicates that 241fake news and 988 real news were predicted correctly.

	0	1
0	185	35
1	241	988

Classification Report and Accuracy

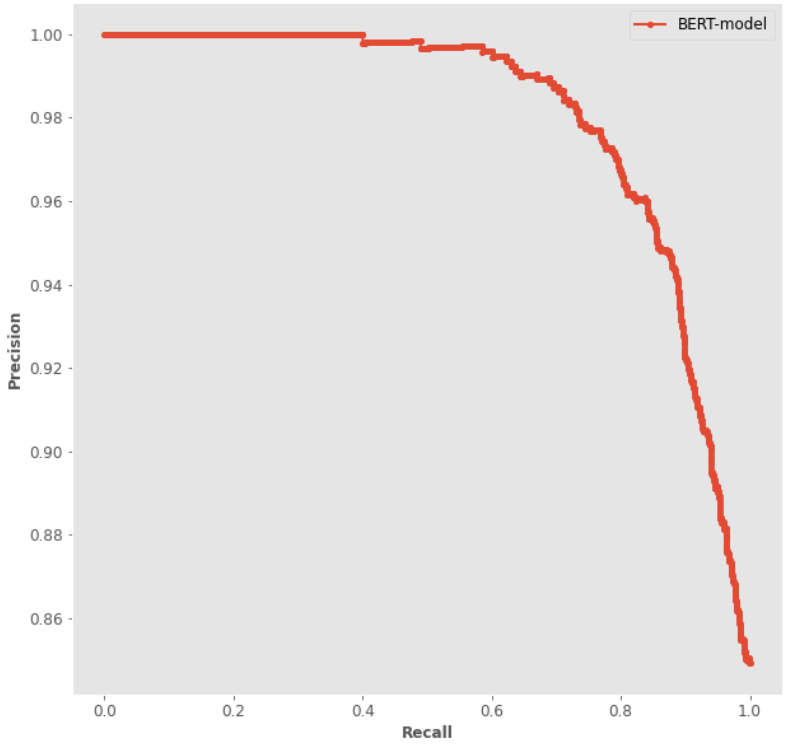
As with the confusion matrix we can see that the accuracy changes drastically between fake and real news. Specifically, it is 43% for fake news and 97% for real news.

	precision	recall	f1-score	support
0	0.43	0.84	0.57	220
1	0.97	0.80	0.88	1229
accuracy			0.81	1449
macro avg	0.70	0.82	0.73	1449
weighted avg	0.89	0.81	0.83	1449

This can be explained by the fact that the titles for fake news use more “real” words whereas in the actual text the diference is more easy to spot.

Precision Recall Curve

Since the accuracy is so different between false and true news, and since there is a big gap of observations between fake and real news (426 fake and 1023 real), we will use Precision Recall Curve instead of Roc curve.



The above Precision Recall Curve show us the **Precision,** defined as the number of true positives over the number of true positives plus the number of false positives and the **Recall**, defined as the number of true positives over the number of true positives plus the number of false negatives.

After considering all the above evaluation metrics for all our different trained models we can conclude that the Feed Forward NN with label encoding is the best. It gives us the highest values in classification report (Accuracy, Recall, f1 Score), ROC Curve score and the minimum cross entropy loss.

Discussion & Future Work

In this project we trained and compared different models. Most of those were very efficient not only because of the pre-processing but also because of the nature of data. We also confirmed that by running dummy classifier which gave an accuracy of 51% which means that our results are far better than those chosen with a not clever way. Of course, there are many cases where the data are more complicated, and the algorithms are not behaving as good. In that situation we can use more data in order for the models to train more efficiently, or try more classifiers like random forest, SVM etc.

In future work, we want to acquire as much data as possible according to this field of study and build again our algorithms and improve our models. While our algorithms perform well on detecting fake news in the training phase, having news from additional sources will improve the accuracy. Later, we want to train algorithms with Greek data and built an app or website and publish the results. Finally, expand Bert model to use the actual text and not just the title.

Members/Roles

We have been working as a team the whole time, but everyone has been assigned as leader in specific tasks.

To begin with, our team consists of people with complementary backgrounds, in order to be more efficient in our project.

1. **Konstantina Georgiopoulou (p2822004):** Data Processing, Validation & Design; she was responsible for collecting, analysing and visualizing the data.
2. **Anastasios Theodorou (p2822007):** Project Management and Designing; this team member was in charge of the ETL process, designing the algorithms as well as overviewing and the completing the whole project (report and presentation), along with its development.
3. **Christos Kallaras (p2822009):** Application Design and Development; he was responsible for designing the application, developing and testing the code and algorithms as well as finding the proper methodology.
4. **Stavros Kasiaris (p2822022):** Modeling and Testing; this team member has executed different types of algorithms, test scenarios and has performed unit tests. He also participated in the report format.

Time Plan

Here we will present our team's time plan:

Tasks - Contents	Start Date	End Date
Phase A		
Introduction (Pre-Analysis)	22-06-21	11-07-21
Vision & Goals		

Methodology		
Members & Roles		
Time Plan		
Data Collection and Analysis	12–07–21	24-07-21
Data Processing	25-07-21	31-07-21
Algorithms, NLP architectures	01-08-21	31-08-21
Phase B		
Results & Quantitative Analysis (incl. visualizations)	23-08-21	31-08-21
Discussion & Future Work	30-08-21	04-09-21
Bibliography		
Appendices		
--> Presentation	04-09-21	07-09-21

Most tasks have been executed in parallel, by multiple people working at the same time (for example “Algorithms, NLP architectures” with “Results & Quantitative Analysis”). It is also worth mentioning, that the majority of the algorithms and the trained models have been done simultaneously. After the training and the completing of one model, we drew its results and its visualizations/plots (especially its evaluation metrics) in order to complete each part in full. Finally, this report has been written in multiple different times based on the process of our python code and its presentation.

Bibliography

For the purpose of this project the slides and the code from the lab of the course Machine Learning and Content Analytics (Part Time) in MScBA, AUEB were extensively used.

Data Collection

https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

https://www.uvic.ca/engineering/ece/isot/assets/docs/ISOT_Fake_News_Dataset_ReadMe.pdf

https://www.uvic.ca/engineering/ece/isot/datasets/fake-news/index.php

Algorithms, NLP architectures

https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/

https://www.v7labs.com/blog/neural-networks-activation-functions

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

https://www.analyticsvidhya.com/blog/2021/06/why-and-how-to-use-bert-for-nlp-text-classification/?fbclid=IwAR0iN7fSpddx979LSgsWfBo6Mkb5bnxVFRR2p713q4FwxDD4HHuGowgjosY

https://towardsdatascience.com/nlp-embedding-techniques-51b7e6ec9f92

https://huggingface.co/bert-base-uncased

https://arxiv.org/abs/1810.04805

https://scikit-learn.org/stable/

https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/