FAKE NEWS ANALYSIS: PREDICTIVE CAPABILITIES AND IMPLEMENTATION ON THE WEB WITH NEURAL NETWORKS

Antonis Gantzos
GREECE

Abstract

The aim of this paper is the implementation of a website capable of distinguishing fake from real news. Firstly, the training of two neural networks was needed, one of which got integrated into the application and makes the predictions using the logistic regression classifier algorithm. In order to train both models, a dataset, containing predetermined values, was used. Afterwards, the performances of both models were compared on various metrics, with the most efficient being incorporated into the website. This process was executed after appropriate preprocessing was performed on the dataset. Python programming language was used for each step described in this research.

1. Introduction to Machine Learning Techniques

1.1 Definition of the Problem

Fake news and misinformation have become a constant problem in modern society, influencing people, communities, and even worldwide events. With the rise of digital news outlets and the ease of information dissemination on the internet, the spread of false news has accelerated, leading to widespread misinformation, confusion, and erosion of trust in traditional sources of information. Fake or unreliable news cover a variety of false or misleading information that are created in order to influence the receivers and prompt them to form opinions according to the narrative that is being propagated. A study conducted by the Electronic Cultural Press Center of the University of Loughborough in 2019 (Chadwick & Vaccari, 2019) revealed that 42.8% of individuals who share news admit to disseminating inaccurate or false information knowingly. This paper aims to create a fake news detection system that can easily be used by the general public to validate a news resource, while simultaneously archiving the fake news it detects, creating a “digital library” of false news. The research can be split into three stages:

1. Introduction to neural networks and fundamental concepts of machine learning and analysis of existing studies that cover similar issues.

2. Implementation of a neural network model capable of making predictions based on a given dataset.
3. Integration of the model into a website and analysis of its capabilities.

1.2. Introduction to Machine Learning and Deep Learning

The basic idea of **machine learning** is that algorithms have the ability to dynamically improve their results with each new execution. Machine learning is categorized into two techniques: *supervised learning*, where the algorithm learns a concept from a given model or dataset, and *unsupervised learning*, where the system is tasked with making associations and grouping records that share common characteristics, thus creating a pattern between them. The effectiveness of a machine learning system is directly dependent on the quantity and quality of the data provided as input into the system's algorithm.

Although **deep learning** is considered a sub-category of machine learning, there are certain points that differentiate the two concepts:

- Training a deep learning algorithm requires a significantly larger amount of data than what machine learning algorithms need to understand the problem and make decisions.
- Deep learning algorithms require much more computational power due to the complexity and volume of data needed in order to train them.
- Deep learning algorithms rely on the creation of neural networks in order to make decisions on their own and perform corresponding actions.

1.3. Introduction to Data Classification

Data classification is a fundamental concept in the field of artificial intelligence and machine learning. It involves training a model to classify incoming data into predefined categories or classes based on their features. There are four data classification tasks in machine learning (Brownlee, 2020):

- **Binary Classification:** Binary data classification tasks involve one class representing the normal condition of given data and another class representing the abnormal condition, for example classifying an email account as "non-spam", which would be the normal condition, in contrast with "spam", which would be the abnormal condition. The class for the normal condition is usually labeled as “0”, and for the abnormal as “1”. Binary classification is often implemented with a model that utilizes the Bernoulli probability distribution to make a prediction.
- **Multi-class Classification:** Multi-class classification refers to classification problems that have more than two class labels. Unlike binary classification, it does not have the notion of normal and abnormal outcomes. Instead, each data entry in a set is categorized into one class from a range of classes based on its features.

- **Multi-Label Classification:** This refers to classification tasks where a data entry in a set can belong to more than one class.
- **Imbalanced Classification:** This refers to classification tasks where the number of data entries in each class varies in size, thus creating an imbalance in the data. Examples include fraud or anomaly detection.

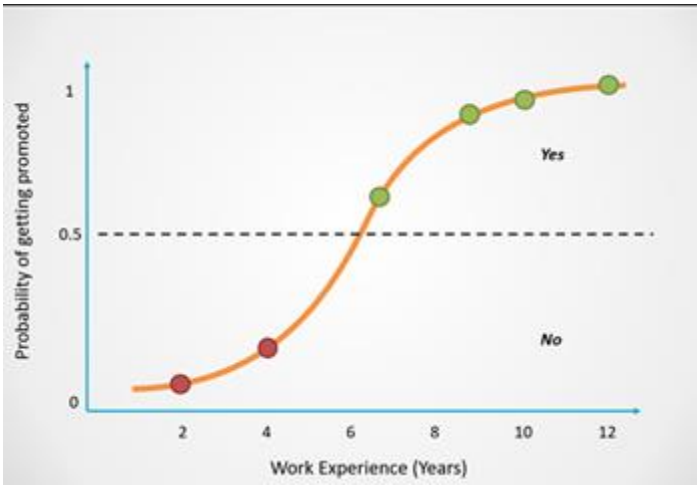
A classification task is handled by implementing a ***classification algorithm***. The most common are Logistic Regression, Support Vector Machines, Decision Trees, K-Nearest Neighors, and ; (Tan et al., 2018):

1.3.1 Logistic Regression

The basic idea behind logistic regression is to use a prediction function that represents the data of entry x and makes a prediction y using the following formula: $P(y=1|x;w) = 1 / (1 + e^{-z})$, where z = g(w^T x), with w^T defined as a vector containing all parameters w for each x (feature) present in the dataset. The result of the function is the probability that y=1, which holds whenever $g(z) \geq 0$. Otherwise, the record's label is classified as 0 (see Figure 1 below).

Figure 1

Data Normalization using Logistic Regression



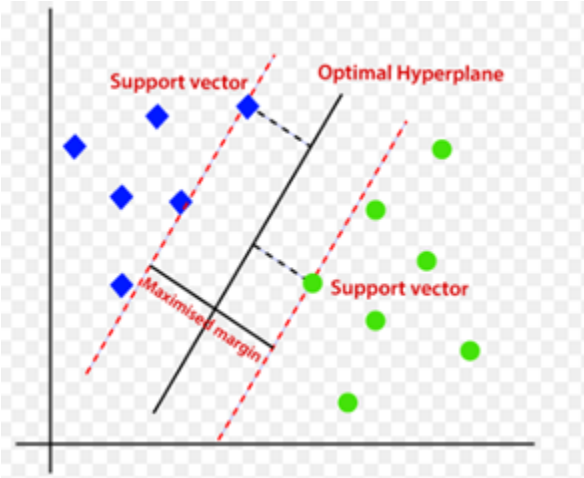
Note. From "*Logistic Regression Explained in 7 Minutes"*, by N. Selvaraj, 2022. (https://www.natasshaselvaraj.com/logistic-regression-explained-in-7-minutes/)

1.3.2 Support Vector Machines

The Support Vector Machine algorithm uses the principle of structural risk minimization to set linear or nonlinear decision boundaries in the space between features, ensuring good performance. Furthermore, it provides very strong adaptation capabilities, controlling the model's complexity to ensure good performance without causing overfitting problems (see Figure 2 below):

Figure 2

Data Classification using SVM



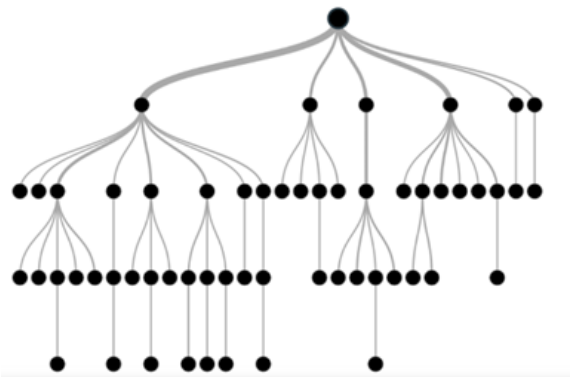
Note. From "*Support Vector Machine Algorithm"* by Javatpoint.com, (n.d.), (https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm)

1.3.3 Decision Tree

The Decision Tree is a non-parametric algorithm used for data classification or regression problems. This means that no assumption about the shapes of the distributions are made, but are approximated by a process of smoothing. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaves. Each node in the tree specifies a test on a data entry, each branch descending from that node corresponds to one of the possible values for that attribute (see Figure 3 below).

Figure 3

Diagram of a Simple Decision Tree



Note. From "Random Forest,GBM(Gradient Boosting Machines)" by C. Kök, 2022 (https://medium.com/@trcahit/random-forest-gbm-gradient-boosting-machines-7cca3badf39b)

1.3.4 K-Nearest Neighbors

K-Nearest Neighbors is a non-parametric algorithm that measures the distance between points to make predictions by identifying the closest "neighbors" of a given point. Working on the assumption that similar points can be found close to each other, the goal of the K-Nearest Neighbors algorithm is to identify the closest "neighbors" of a given data entry and assign a class label to it. The *Euclidean Distance formula* is commonly used to measure the distance of a given point from its neighbors (see Figure 4 below).

Figure 4

K-Nearest Neighbors algorithm example

Note. From "*What is the k-nearest neighbors (KNN) algorithm?*", by IBM, (n.d.) (https://www.ibm.com/topics/knn)

1.3.5 Naive Bayes

Based on Bayes' theorem, the Naive Bayes algorithm assumes independence between features and calculates the probability that a given input instance belongs to a specific class.

1.4. Introduction to Neural Networks

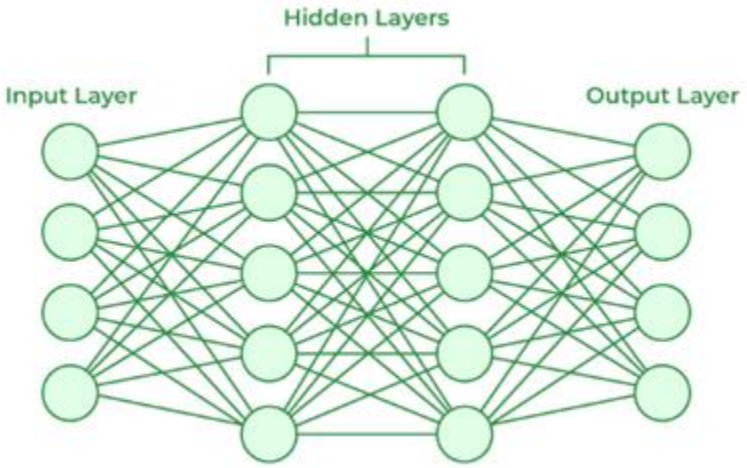
A **neural network** is a method in artificial intelligence that teaches computers to process data in a manner similar to the human brain. It is a type of deep learning process that uses interconnected nodes (**neurons**) in a layered structure resembling the human brain. This creates an adaptive system that a model can use to learn from its mistakes and continuously improve on its predictions (Amazon Web Services, n.d.). A neural network can be structured into three basic layers:

1. **Input Layer.** Input data enters an artificial neural network through the input layer. The input nodes process, analyze, or categorize the data and pass it to the next layer.
2. **Hidden Layers.** Data is received from an input layer or other hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it to the next layer.

3. **Output Layer.** This provides the final outcome of all data processing done by previous layers. It can have a single or multiple nodes, depending on the classification task the neural network is working on. The architecture of a neural network can be seen in Figure 5

Figure 5

Architecture of a Neural Network



Note. From "Building a Basic Neural Network from Scratch: A Step-by-Step Guide", by D. Bhatnagar, 2023, *Medium*, https://bhatnagar91.medium.com/building-a-basic-neural-network-from-scratch-a-step-by-step-guide-7a6f97979ddd/

Having covered the basic ideas behind neural networks, all that remains is to answer the following question: How is a neural network trained? In supervised learning, artificial neural networks are given labeled datasets that provide the correct answer in advance and gradually build knowledge step by step by making repeated predictions on these datasets. Once the network is trained, it begins to make estimates about data it has never processed before.

1.5 Related Research

In related research, the following articles propose various approaches for detecting fake news using a variety of machine learning techniques.

Sanei et al. (2017), have explored ways to increase the efficiency of the KNN algorithm to provide better results for their model's predictions. The algorithm is used to select the best parameters for the nonlinear functions that are most suitable for each feature, with results being generally better compared to those in a similar study conducted Nair and Kashyap (2019), who proposed the use of resampling and interquartile range (IQR) techniques in the data preprocessing steps, which are normalized for better algorithm performance.

Kesarwani et al. (2021) created a fake news detection model based on news headlines, as well as various articles and news frequently appearing on several users' social networks.

Nagashri and Sangeetha (2021) attempted to identify fake news using evaluation metrics techniques and tested many machine learning concepts based on accuracy, precision, recall, and F1 score, concluding that the TFIDF vectorizer should be the most preferred text preprocessing technique. Further reference to the metrics used to measure the performance of a model will be made in section 2 of this paper.

Vijayaraghavan et al. (2020) attempted to define a connection between words and the context in which they appear within the text, as well as how they could be used to categorize a given news article as true or fictional. They used models such as Count Vectorizer to convert texts into numerical representations and investigated which model is capable of more accurately determining the article as real or fake.

2. Fake News Detection System

2.1 Introduction

This section of the paper describes the methodology and steps followed to implement and train a neural network model. Initially, a training dataset was used, to which various natural language processing techniques were applied. Subsequently, two neural network models are created: a linear and a non-linear. The two models were compared based on certain metrics, and the model that produced the greater results was integrated into the website.

2.2 Dataset Description

The dataset used is sourced from the GitHub repository (Lifferth, 2018) and consists of a collection of news articles. Each news article corresponds to a record in the dataset, with the following features:

- **ID:** The record number.
- **Title:** The title of the news article.
- **Author:** The name of the author who wrote the article text.
- **Text:** The text of the news article.

Each record also includes a label, which can take two values: 0 or 1. This value assists in categorizing a news article as true, if the value is “0”, or false, if the value is “1”. The dataset comprises approximately 10,000 records.

For the model implementation, Python was utilized in conjunction with the PyTorch library for creating the neural network. Pandas, Scikit-learn, and

NumPy were used for data preprocessing and result evaluation. A general format of the dataset is seen in Figure 6.

Figure 6

A general format of the dataset

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

2.3 Data Pre-processing

Due to the natural language from which the texts representing the data are composed, some noise must be removed before data is fed into a neural network. In order to make input data suitable for the algorithms to operate, it undergoes various pre-processing techniques aimed at smoothing the dataset. These techniques are as follows:

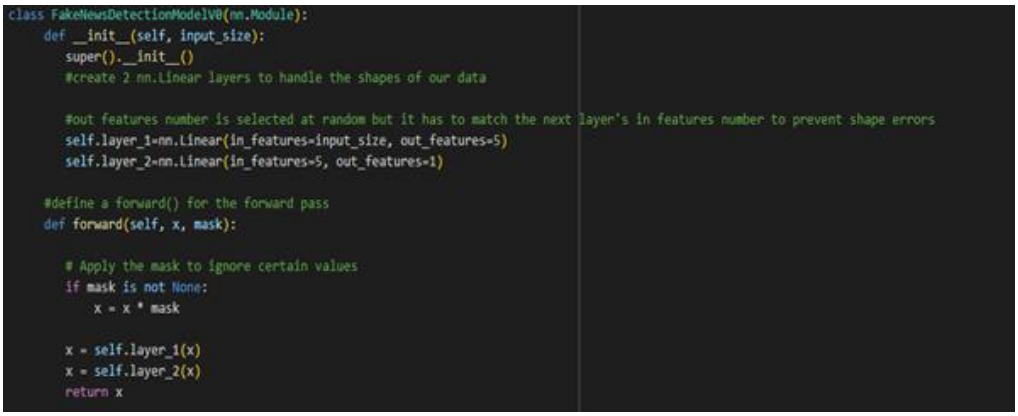
- **Removal of duplicate and missing records:** Entries that appear more than once in the dataset and entries that do not provide any information that can be utilized later are removed.
- **Conversion of all letters to lowercase:** This technique aids in uniformity among the dataset's entries and facilitates their processing.
- **Removal of punctuation marks from each entry:** Performed for similar reasons as lower case conversion.
- **Removal of unnecessary words in each entry of the set (Stopwords removal):** Stopwords are essentially words that add value to other words or define a relationship between words. These may include various adjectives, adverbs, prepositions, conjunctions, and pronouns. Since the dataset includes various articles, it is necessary to remove these words before data is provided as input.
- **Use of Count Vectorizer:** The final step in data preprocessing is to convert the text given as input to the model into a vector. Vectorization is a process of converting words in a text to a form that can be read by a machine. Therefore, before being given as input to the model, all texts must be represented as vectors (tensors). Each value of the vector represents the frequency of each word in the text. For performing this conversion, the capabilities of a Count Vectorizer are utilized, which, for a given text, creates an array where each position represents a unique word in the text, and the corresponding value of that position represents the frequency of the word (Jain, 2021).

2.4 Model Implementation

The next step is to implement a model of a neural network, which is observed in Figure 7. This model was implemented based on the architecture of neural networks analyzed in the previous section. The flow can be clearly observed in the image below. In **`__init__()`** function, the 2 layers that will process the input data are initialized. The actual processing is executed in the **`forward()`** function. For the hidden layers, the number hidden neurons that will perform the computations is set as relatively small, considering the simplicity and volume of the data, while the number of features that will be inputted is set as one, as well as the number of features that will be produced as output, since the goal is to have one result for each news article (0 for true news or 1 for fake news).

Figure 7

Neural Network Model



A similar process will be followed for the development of the second model, which, unlike the current one that employs a **Linear** approach, utilizes **Non-Linearity** for processing. However, before continuing with the training of both models, it is necessary to clarify the process followed by each of the two approaches mentioned, as well as their differences.

2.5 Linearity vs Non-Linearity

There are two types of neural network models: linear and non-linear.

2.5.1 Linearity

In many cases, linearity is the simplest and most effective approach. A linear model, essentially, fits a straight line to the data, allowing it to make predictions based on a linear relationship between the input features and the output variable. For this process, the *linear regression* function is utilized: $y=b0+b1 \cdot x1$, where y is the dependent variable (prediction outcome), $x1$ is the independent variable or feature, $b0$ is the intercept of y with the y-axis (constant), and finally $b1$ is the slope coefficient.

2.5.2 Non-Linearity

Non-linear neural network models can take many forms, from polynomial models that fit curves to the data, to neural networks that can learn complex patterns in high-dimensional data. Non-Linear models are often more powerful their Linear counterpart, because they can recognize more complex relationships between variables. In a classification problem, non-linear models can identify more complex than linear decision boundaries that define different classes. However, they may be more challenging to use than linear models, as they usually require much more training data and computational power to achieve good results (Zarra, 2023).

2.6 Training the Models

The next step after preprocessing the data is implementing the training process. A **train_test_split** method is used to split the dataset into two parts. It is defined that 80% of the records of the dataset will be used for training the algorithms, and the remaining 20% will constitute the test sample size. All columns of the dataset are concatenated, forming a unified column, and the data is then split and passed to the Count Vectorizer.

For training a neural network model, the following concepts need to be introduced and implemented (Patil, 2023):

- **Loss Function**: Also known as a cost function, a loss function is used to measure the accuracy of a model's predictions. It calculates the difference between the predicted output and the actual output for each training sample. The goal of the model is to minimize the loss function by the end of training. The smaller the loss function, the better and more accurate the model's parameter set will be in producing predictions.
- **Optimizer**: The optimizer adjusts the model's parameters to gradually minimize the loss function. It's worth noting that optimizers can also adjust various hyperparameters, such as the learning rate, momentum, and others. The main optimizers include **Gradient Descent (GD)**, **Stochastic Gradient Descent (SGD)**, and **Adaptive Moment Estimation (Adam)**. In this particular study, it was determined that the Stochastic Gradient Descent optimizer was better suited for training the models.

The process of training the models involves the following steps:

- Input data is passed through the **forward()** function for processing, done by the hidden layers of the network, and the model makes a prediction on it.
- The loss function is computed, estimating how close the model's prediction was to the actual label of the data. The desired output is the gradual minimization of loss function, which indicates good performance.

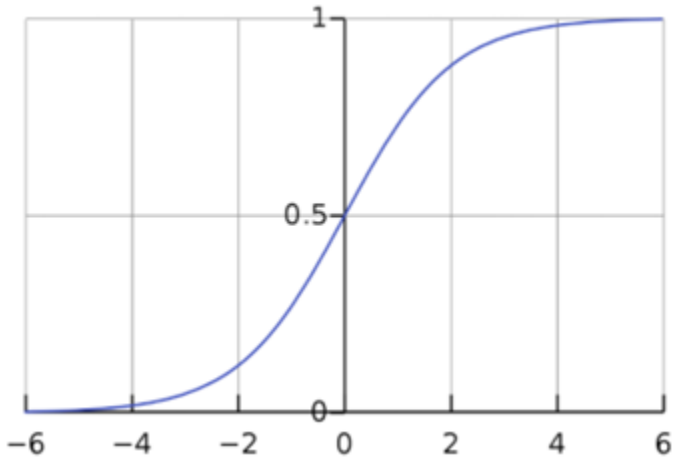
- The Stochastic Gradient Descent (SGD) algorithm is used to improve the model's parameters, so that the next prediction is more accurate than the previous one in order to make the loss function decrease.
- The process is repeated for a certain number of *epochs*, providing the models with enough opportunities to learn from their mistakes and improve their performance. The goal is for the loss function of each model to reach as close to zero as possible.

Each iteration recalculates the evaluation metrics of the models, which will be discussed in more detail below.

It's also worth noting the process of converting the model's output values into the desired output values, namely “0” or “1”. The values output by the model are called **logits**, and they are the values generated by a neural network before applying an activation function. They are the non-normalized probabilities of a data entry to belong to a certain class. Converting logits into probabilities makes it easier to understand the final output of the neural network for each prediction. The activation function chosen for this conversion is the **sigmoid function,** which can be seen in Figure 8 below. As shown in the diagram below, this function, after converting logits into probabilities, applies the following rule: If the probability is greater than or equal to 0.5, then the output value is 1; otherwise, the output value is 0. Therefore, the output given from both models is the probability of a news item being fake.

Figure 8

Sigmoid Function



Note. From “Sigmoid function”, *Wikipedia* (https://en.wikipedia.org/wiki/Sigmoid_function)

2.7 Evaluation Metrics

After completing the training, it is important to highlight the generated results. The metrics of *Accuracy*, *Precision*, *Recall*, are used to measure the performance of the models (Agrawal, 2024).

2.7.1 Accuracy

Accuracy measures how often the classifier predicts correctly. We can define accuracy as the ratio of the number of correct predictions to the total number of predictions. Correct predictions are those categorized correctly as 0 (**True Positives or TP**) and those categorized correctly as 1 (**True Negatives or TN**), while incorrect predictions are data points that, although their true output value should be 1, the model predicts as 0 (**False Positives or FP**), or vice versa (**False Negatives or FN**). The combination of these four categories constitutes the total predictions of the model. Its formula can be seen in Figure 9 below.

Figure 9

Formula for calculating the Accuracy metric

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Note. From "Metrics to Evaluate your Classification Model to take the right decisions" by S. K. Agrawal, 2024, *Analytics Vidhya*, https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/)

2.7.2 Precision

Precision is used to measure how many of the correctly predicted instances actually turned out to be positive. It is useful in cases when a False Positive is more concerning than a False Negative. Its formula can be seen below.

Figure 10

Formula for Calculating the Precision Metric

$$\text{Precision} = \frac{TP}{TP + FP}$$

Note. From "F1 Score in Machine Learning: Intro & Calculation" by R. Kundu, 2024, *V7labs* (https://www.v7labs.com/blog/f1-score-guide)

2.7.3 Recall

Recall explains how many of the actual positive instances were correctly predicted by the model. Recall is a useful measure in cases when a False Negative is more concerning than a False Positive. Its formula can be seen in Figure 11 below.

Figure 11

Formula for Calculating the Recall metric

$$\text{Recall} = \frac{TP}{TP + FN}$$

Note. From "F1 Score in Machine Learning: Intro & Calculation" by R. Kundu, 2024, *V7labs* (https://www.v7labs.com/blog/f1-score-guide)

2.7.4 F1 Score

An F1 Score is the average of Precision and Recall. It is commonly used when there is a need for a balanced use of both factors in a model evaluation. Its formula can be seen in Figure 12 below.

Figure 12

Formula for calculating the F1 metric

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Note. From "F1 Score in Machine Learning: Intro & Calculation" by R. Kundu, 2024, *V7labs* (https://www.v7labs.com/blog/f1-score-guide)

Regarding the dataset in this study, it is observed from Figure 13 that the entries constituting the dataset are almost equally distributed. In such a case where there is no issue of imbalanced classes, the most important metric to consider is Accuracy, as well as the loss function results of each model.

Figure 13

General dataset shape

```
the shape of our data when duplicates are removed is (10800, 5)
1    5417
0    5383
```

2.8 Model Comparison Review

This section compares the performance results of the linear and non-linear models.

2.8.1 Linear Model

The performance results of the linear model can be observed from the following diagrams, beginning with Figure 14.

Figure 14

Performance Measurement Results of the Linear Model over Five Epochs

```
  3%|▌         | 1/30 [00:01<00:36,  1.27s/it]
epoch : 0, training loss: 0.71091, test loss: 0.58467, training accuracy 0.429, testing accuracy
0.666, training precision: 0.43095, training recall: 0.98502, training F1: 0.59958,testing
precision: 0.87302, testing recall: 0.25701, testing F1: 0.39711
 20%|██        | 6/30 [00:05<00:21,  1.10it/s]
epoch : 5, training loss: 0.43858, test loss: 0.42515, training accuracy 0.89075, testing accuracy
0.887, training precision: 0.88005, training recall: 0.86636, training F1: 0.87315,testing
precision: 0.86041, testing recall: 0.87850, testing F1: 0.86936
 37%|███▋      | 11/30 [00:10<00:16,  1.12it/s]
epoch : 10, training loss: 0.36925, test loss: 0.36675, training accuracy 0.9107499999999998,
testing accuracy 0.9010000000000001, training precision: 0.90871, training recall: 0.88306,
training F1: 0.89571,testing precision: 0.87991, testing recall: 0.89019, testing F1: 0.88502
 53%|█████▎    | 16/30 [00:15<00:13,  1.04it/s]
epoch : 15, training loss: 0.32381, test loss: 0.32917, training accuracy 0.925, testing accuracy
0.912, training precision: 0.92789, training recall: 0.89689, training F1: 0.91213,testing
precision: 0.90094, testing recall: 0.89252, testing F1: 0.89671
 70%|███████   | 21/30 [00:19<00:08,  1.12it/s]
epoch : 20, training loss: 0.28916, test loss: 0.30171, training accuracy 0.934, testing accuracy
0.9179999999999999, training precision: 0.93914, training recall: 0.90668, training F1:
0.92263,testing precision: 0.90421, testing recall: 0.90421, testing F1: 0.90421
 87%|████████▋ | 26/30 [00:23<00:03,  1.11it/s]
epoch : 25, training loss: 0.26075, test loss: 0.28024, training accuracy 0.9395, testing accuracy
0.924, training precision: 0.94411, training recall: 0.91475, training F1: 0.92920,testing
precision: 0.91315, testing recall: 0.90888, testing F1: 0.91101
100%|██████████| 30/30 [00:27<00:00,  1.07it/s]
total training time is 28.001 seconds
```

It is observed that both in the training data and the test data, the loss function steadily decreases as the model learns from its previous predictions, while accuracy reaches nearly 95% correct predictions with the other metrics following closely behind with slightly lower percentages. Additionally, it is considered positive that the measurement results in the test data do not differ much from those in the training data, as this indicates that the classifier is not overfitting the training data to the point where it can only make predictions on them and no other dataset that it has no prior knowledge on, as seen in Figures 15 and 16 below.

Figure 15

Diagrammatic Representation of Linear Model Results

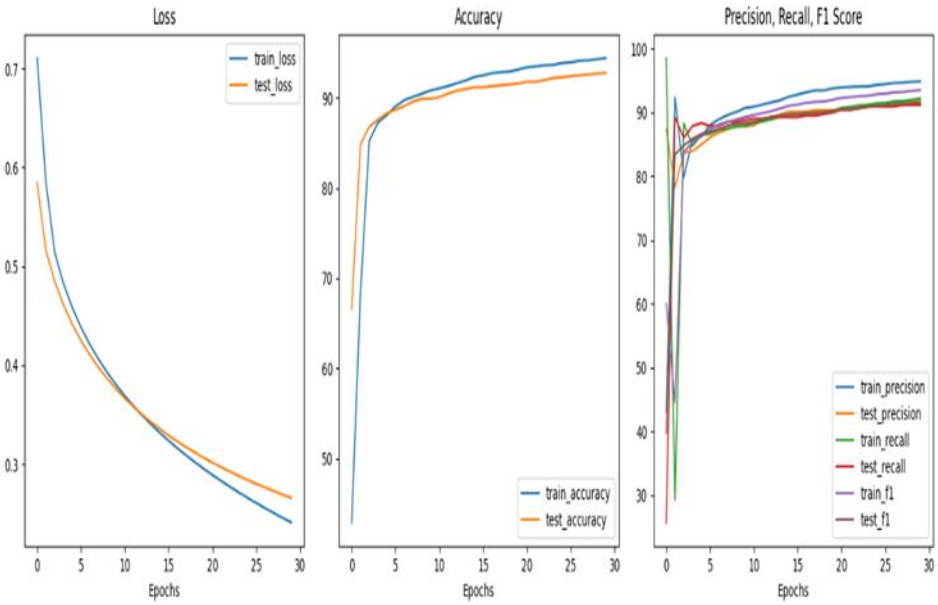
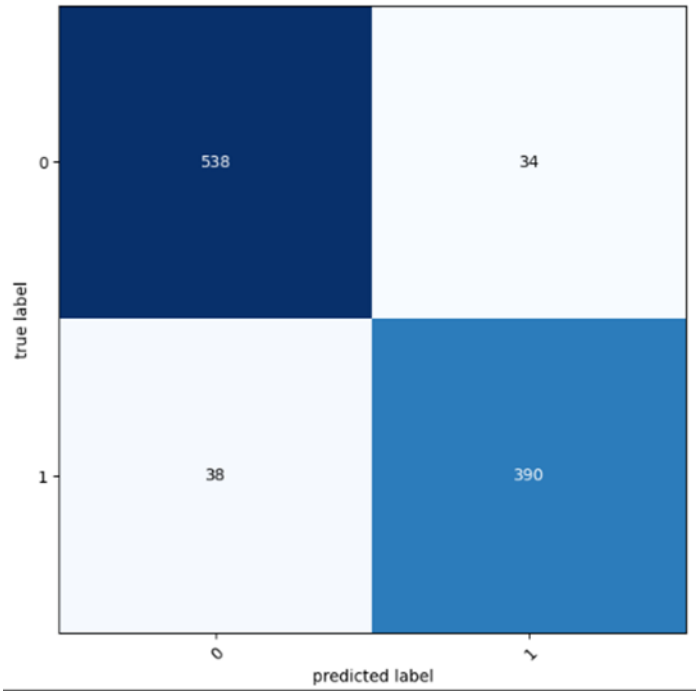


Figure 16

Accuracy Visualization of Linear Model Results using Confusion Matrix



2.8.2 Non-Linear Model

The performance results of the non-linear model can be observed from Figures 17 and 18. It is observed that both in the training data and the test data, the loss function remains steady throughout the model training process, as does the prediction accuracy, with a percentage of approximately 43%. Since it performs binary classification, this percentage is worse than even a random guess.

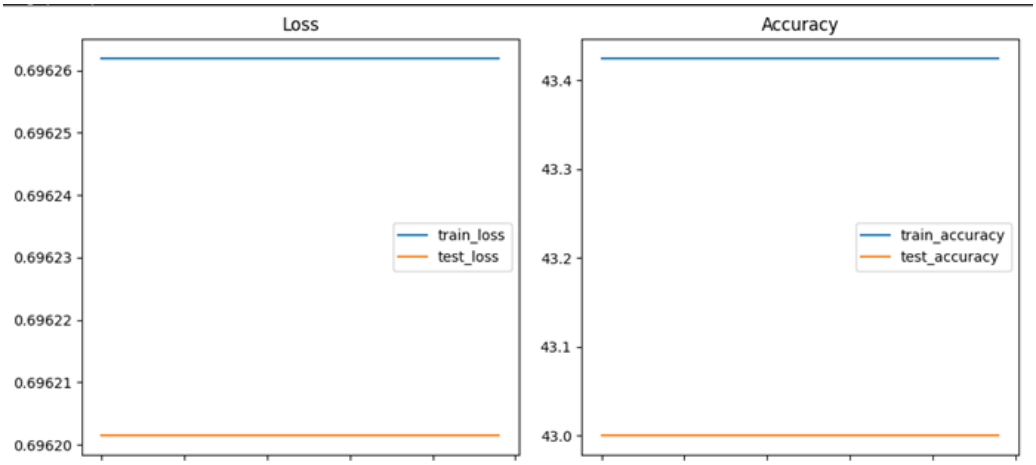
Figure 17

Results of Non-linear Model Performance Measurements per Five Seasons

```
 4%|▌         | 1/25 [00:01<00:41,  1.74s/it]
epoch : 0, training loss: 0.69626, test loss: 0.69620, training accuracy 43.425000000000004,
testing accuracy 43.0
 24%|■■▌       | 6/25 [00:11<00:35,  1.87s/it]
epoch : 5, training loss: 0.69626, test loss: 0.69620, training accuracy 43.425000000000004,
testing accuracy 43.0
 44%|■■■■▌     | 11/25 [00:22<00:28,  2.06s/it]
epoch : 10, training loss: 0.69626, test loss: 0.69620, training accuracy 43.425000000000004,
testing accuracy 43.0
 64%|■■■■■■▌   | 16/25 [00:32<00:18,  2.03s/it]
epoch : 15, training loss: 0.69626, test loss: 0.69620, training accuracy 43.425000000000004,
testing accuracy 43.0
 84%|■■■■■■■■▌ | 21/25 [00:38<00:04,  1.24s/it]
epoch : 20, training loss: 0.69626, test loss: 0.69620, training accuracy 43.425000000000004,
testing accuracy 43.0
100%|■■■■■■■■■■| 25/25 [00:42<00:00,  1.72s/it]
total training time is 42.935 seconds
```

Figure 18

Diagrammatic Representation of Non-linear Model Results



It is concluded from the superior performance of the linear model, as documented above, that the linear model will ultimately be integrated into the website. Further analysis will follow in the next section to showcase it in more detail.

3. Fake News Detection Application

3.1 Introduction

In the previous section, the methodology for creating and training a neural network model to make predictions was discussed. In this section, the focus of the study will be the implementation and functionalities of a website that integrates the aforementioned model. This website allows users to make predictions and effectively verify the validity of the information they consume. These predictions are stored in a database, which functions as a "digital fake news repository," providing a dataset that can be utilized by anyone conducting a similar research.

3.2 Application technologies

- **React.js | Frontend framework:** Used on all application pages related to the User Interface for user interaction with the application and its functionalities.
- **Python Flask | Backend framework:** Used to create the server and for the interaction of the website with the database.
- **MySQL Database | Relational Database**

3.3 Database Analysis:

The database consists of a single table, which includes the following fields:

- **ID**: Unique attribute of each record in the database
- **Article**: Text describing a news article
- **Fake Probability**: Probability that the news article should be considered false information
- **Label**: Takes only 2 values, “0” or “1”, depending on whether a news source is true or false.

3.4 Application Functionalities

The website performs three basic functions: News Prediction, Display of Fake News, and Database Data Export.

3.4.1 News Prediction

News prediction is the most basic function of the application. Since the model has been integrated into the website, a user interface is implemented for users to utilize its prediction capabilities. A text area is created where the user enters the text of an article they want to verify, and by pressing the Predict button on the left, the server response appears, indicating whether the news is reliable or not, along with the probability of it being fake news. It is noted here that since

news labeled “0” is defined as true and news labeled “1” is defined as false, the closer the probability received by the user is to 1, the more likely it is that the article should be considered false information. Additionally, by pressing the second button on the right, brief instructions on the application's functionality are provided, as well as contact information to enable users to provide feedback for possible future improvements. An example prediction result can be seen in Figure 19.

Figure 19

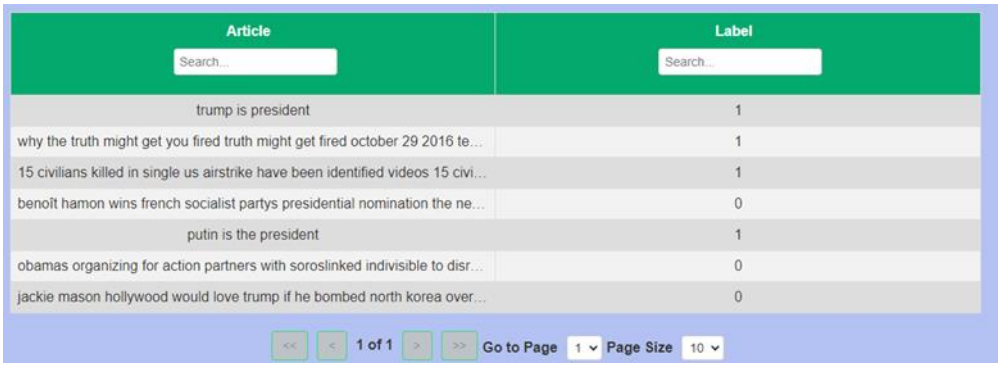
Prediction Results

3.4.2 Display of Fake News Database

After each prediction, the news article provided by the user is inserted into the table analysed in the previous section, provided that the news has not been previously entered, ensuring each record is unique. The probability of the news being fake, as well as its label, is also stored. The table, which is seen in Figure 20, is presented on the website using the React framework. The collection of news articles is organized into table pages, using a custom paginator. Additionally, a search filter is provided so that users can research specific news articles.

Figure 20

Data display in table

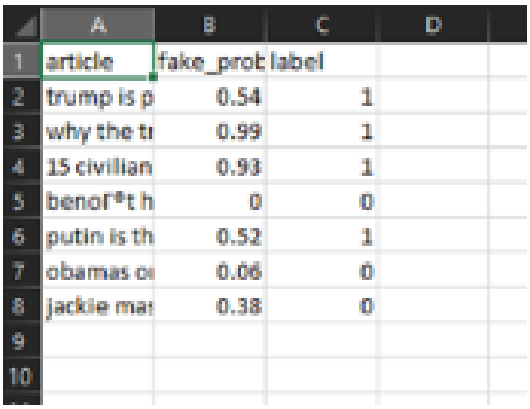


3.4.3 Database Data Export

Finally, the website allows the user to export the stored data from the database, with the aim of creating a new dataset that can be used in similar studies. This functionality of the website is activated by pressing the button shown below. The result is presented in Figures 21 and 22; the first 7 data rows in Figure 21 correspond to the seven articles in Figure 20.

Figure 21

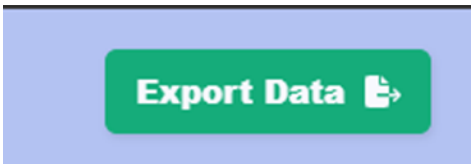
Export Results



	A	B	C	D
1	article	fake_prob	label	
2	trump is p	0.54	1	
3	why the tr	0.99	1	
4	15 civilian	0.93	1	
5	benof*t h	0	0	
6	putin is th	0.52	1	
7	obamas o	0.06	0	
8	jackie ma	0.38	0	
9				
10				

Figure 22

Export button



3. Conclusion

This paper focuses on detecting fake news on the internet using machine learning. A neural network, implementing binary classification on data, was developed, labeling reliable information sources with “0” and unreliable sources with “1”.

From the literature review, it was found that the best approach to the problem is binary classification. During the implementation methodology, two neural network models were created: one linear and one non-linear. After preprocessing the input data, both models underwent the exact same training process. Their performance was compared under the same conditions, using metrics such as prediction accuracy and recall. The linear model outperformed the non-linear one and was integrated into the website to allow users to check their information sources as effectively as possible.

The study concludes that, although the non-linear approach is usually considered to produce better results for applications dealing with simpler datasets with less complexity and fewer dimensions, the linear approach is more likely the best solution due to the utilization of Linear Regression, which is more effective for datasets with a smaller number of dimensions.

References

Agrawal, S. K. (2024, February). Metrics to evaluate your classification model to take the right decisions. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

Amazon Web Services. (n.d.). *What is a neural network?* https://aws.amazon.com/what-is/neural-network/

Bhatnagar, D. (2023). Building a Basic Neural Network from Scratch: A Step-by-Step Guide. *Medium.* https://bhatnagar91.medium.com/building-a-basic-neural-network-from-scratch-a-step-by-step-guide-7a6f97979ddd/

Brownlee, J. (2020, November). 4 types of classification tasks in machine learning. *Machine Learning Mastery*. https://machinelearningmastery.com/types-of-classification-in-machine-learning/

Chadwick, A., & Vaccari, C. (2019). *News sharing on UK social media: Misinformation, disinformation, and correction survey report*. Loughborough University. Report. https://hdl.handle.net/2134/37720

IBM. (n.d.). *What is the k-nearest neighbors (KNN) algorithm?* https://www.ibm.com/topics/knn

Jain, P. (2021, May). Basics of count vectorizer. *Towards Data Science*. https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c

Javatpoint. (n.d.). *Support vector machine algorithm.* https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm)

Kesarwani, A., Chauhan, S. S., Nair, A. R., & Verma, G. (2021). Supervised machine learning algorithms for fake news detection. In *Advances in Communication and Computational Technology: Select Proceedings of ICACCT 2019* (pp. 767-778). Springer Singapore. https://doi.org/10.1007/978-981-15-5341-7_58

Kök, C. (2022). Random Forest,GBM(Gradient Boosting Machines). https://medium.com/@trcahit/random-forest-gbm-gradient-boosting-machines-7cca3badf39b

Kundu, R. (2024). F1 Score in Machine Learning: Intro & Calculation. *V7labs.* https://www.v7labs.com/blog/f1-score-guide

Lifferth, W. (2018). Fake News. Kaggle. https://kaggle.com/competitions/fake-news

Nagashri, K., & Sangeetha, J. (2021). Fake news detection using Passive-Aggressive Classifier and other machine learning algorithms. In S.M. Thampi, E. Gelenbe, M. Atiquzzaman, V. Chaudhary, & KC Li, (Eds.) *Advances in computing and network communications: Lecture notes in electrical engineering, 736*. (pp. 221-233). Springer, Singapore. https://doi.org/10.1007/978-981-33-6987-0_19

Nair, P., & Kashyap, I. (2019). Hybrid pre-processing technique for handling imbalanced data and detecting outliers for KNN classifier. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)* (pp. 460-464). IEEE. doi: 10.1109/COMITCon.2019.8862250

Patil, S. U. (2023, January). Loss functions and optimizers in ML models. *Medium*. https://medium.com/geekculture/loss-functions-and-optimizers-in-ml-models-b125871ff0dc

Sanei, F., Harifi, A., & Golzari, S. (2017). Improving the precision of KNN classifier using nonlinear weighting method based on the spline interpolation. In *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)* (pp. 289-292). IEEE. doi: 10.1109/ICCKE.2017.8167893

Selvaraj, N. (2022). *Logistic Regression Explained in 7 Minutes*. https://www.natasshaselvaraj.com/logistic-regression-explained-in-7-minutes/)

Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). *Introduction to data mining* (2nd ed., pp. 147-174). Pearson.

Vijayaraghavan, S., Wang, Y., Guo, Z., Voong, J., Xu, W., Nasseri, A., Cai, J., Li, L., Vuong, K., & Wadhwa, E. (2020). Fake news detection with different models. *arXiv preprint arXiv:2003.04978*. Cornell University. https://doi.org/10.48550/arXiv.2003.04978

Zarra, R. (2023, March). Machine learning: Linearity vs non-linearity. *LinkedIn*. https://www.linkedin.com/pulse/machine-learning-linearity-vs-nonlinearity-reday-zarra/

Author Details

Antonis Gantzos
GREECE
antonisgantzos@gmail.com

Editors' note: This is a pre-publication copy of the paper and intentionally does not include page numbers, which will be included in the complete proceedings of ICICTE 2024.