THE UNIVERSITY OF SYDNEY

Project Report for ENGG2112

Towards Robust Fake News Detection: A Comprehensive Machine Learning Study

Lam Duy Nhat Le, 510670000, Software Engineering
Ruei En Tan, 520281642, Software Engineering
Yijiang Luo, 520068607, Software Engineering
Shuaiyu Lu, 520381447, Software Engineering

Faculty of Engineering

February 16, 2024

Executive Summary

In an era where fake news undermines public discourse, this paper investigates the efficacy of Machine Learning (ML) techniques to differentiate between fake and genuine news articles. Utilizing a dataset characterized by titles, authors, content, and veracity labels, we engaged in rigorous preprocessing and implemented feature extraction methods such as TF-IDF and One-Hot-Encoding. Four AI models—Naive Bayes, K Nearest Neighbours, multilayer perceptron and dense layer Neural Network— were trained and exhibited high accuracy on the training data, yet faced a large decline in performance on an unseen dataset, revealing the inherent challenge of relying solely on text patterns for fake news detection. This study not only evaluates the capabilities of various ML models in identifying disinformation but also emphasizes the necessity of cross-verification tools and the collective responsibility of individuals, news outlets, and social platforms in the fight against fake news proliferation. Our research delineates a path forward that combines technological, social, and ethical approaches to enhance the resilience of news verification systems.

Contents

1 Background and Motivation

In the age of information, fake news have become even easier to disseminate through websites, tv, and videos. People largely obtain information through the internet evident by the fact that "two thirds of the global population is connected to the world wide web." Whether it be news, entertainment, or just plain old googling things, we have become extremely reliant on technology as a source of our information. Although the adage "don't trust everything on the internet" is common knowledge, people have continuously been tricked and misled. In recent years, with the advancement of machine learning, misinformation has never been easier to spread.In fact, according to research, "more than 40% of the news shared on social media platforms was found to be fake". As ML tools can generate hundreds of fake news stories, memes, and even videos of politicians speaking. As such we aim to create a model that can discern whether an article

For this research, we have chosen a dataset provided by the UTK Machine Learning Club and made available on Kaggle as part of a competition. We opted for this dataset for several reasons:

- Size: With 20,000 entries, the dataset is sufficiently large to train and test machine learning models effectively.
- Cleanliness: The dataset is well-curated, reducing the amount of preprocessing required, which is often a time-consuming step in machine learning projects.
- Comprehensiveness: It includes multiple features such as the text, title, and author, along with labels indicating whether the news is fake or not. This allows for a multi-faceted approach to the problem.

Through this research, we hope to make a meaningful contribution to the ongoing efforts to combat misinformation, thereby fostering a more informed and trustworthy digital ecosystem.

2 Objectives and Problem Statement

The problem this research paper aims to solve is to identify whether or not machine learning models are accurate in predicting the trustworthiness of news articles. To achieve this, we aim to utilize feature extraction and machine learning models. Specifically, we plan to use Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction, while for model training we intend to employ four different machine learning algorithms: Naive Bayes Classifier, K-Nearest Neighbour, and neural networks such as Multi-layer Perception and Dense Layer. The aim of using multiple machine learning models is to examine whether specific machine learning models are better for classifying articles. Naive Bayes and K-Nearest Neighbour are good at sorting text into different categories [3]. Neural Networks offer the advantage of learning complex patterns in the data, which could be particularly useful in fake news detection. The two different neural networks used come from two different libraries, one is from SKlearn, and the other is from Keras, with Keras having the advantage being able to take advantage of advanced processes such as various activation features, as well as being able to utilize GPU.

3 Methodology

3.1 Data Pre-Processing

While the dataset this project utilizes excels in cleanliness, volume, and comprehensiveness, it still contains issues such as having "too much data," "too little data," and "fractured data" [1] [1]. Each of these problems presents unique challenges that necessitate tailored solutions. Specifically, too much data arises from an abundance or a surplus of irrelevant data; the sheer volume can be overwhelming, making the analysis process cumbersome and time-consuming; irrelevant data can cause misinterpretations or inaccurate conclusions. On the flip side, having too little data is equally problematic as it may result in under-fitting, where the model fails to capture the underlying trends, compromising the accuracy of predictions. Finally, the fractured data problem arises from the diverse origin of data, which could cause the data to be incompatible. In light of these challenges, cautious data preprocessing must be employed to rectify these issues, enhancing the dataset's readiness for practical machine learning training and subsequent analytical endeavors.

First, let us examine the missing data indicators depicted in Table

1.

As observed, this dataset exhibits a problem; specifically, it has missing attribute values in the title, author and text column. Generally, we have two options: to remove or retain these entries with missing values. However, our situation does not permit the removal of any data, as this would result in the deletion of more than 10% of our data, which is substantial. This notion aligns well with the proposed line of thought in the "Data Preprocessing and Intelligent Data Analysis" journal, as "the total amount of data may not be sufficient. On the other hand, the remaining values in the data record may contain very useful information." However, we will not retain the dataset as is because null values manifest in various forms. Therefore, it is prudent to fill these values with an easy-to-understand placeholder like "Unknown" or "Ambiguous." This approach aims to ensure a level of consistency and completeness in the dataset, facilitating more accurate and meaningful analysis. By replacing the missing values with a discernible placeholder, we strive to maintain the integrity of the data while making the absence of information explicit and easily identifiable.

After addressing missing values, navigating the subsequent diagnostic steps related to author identification can pose a challenge due to the inherent uncertainty regarding where issues might emerge within the dataset. Upon examining the dataset, we discovered that some authors are ambiguously mentioned.

Entries such as "admin" or the website's name often indicate a lack of human authorship behind the news writing, potentially rendering the information unreliable. As illustrated in Figure 2 (which showcases the distribution of ambiguous author entries), this scenario exemplifies a noisy data problem, as it introduces irrelevant or misleading information into the dataset. The ambiguity in authorship can complicate the analysis, making it difficult to ascertain the reliability and authenticity of the news articles in question. This form of noise could significantly impact the integrity of the analyses and the machine learning models trained on this data. For instance, a model tasked with identifying reliable news sources may erroneously categorize these ambiguously authored entries. This, in turn, undermines the model's accuracy and effectiveness. Therefore, renaming these values as "Unknown" is a crucial step, ensuring consistency with previous data preprocessing steps and making the authorship ambiguities easier to identify.

3.2 Feature Extraction

3.2.1 TF-IDF

TF-IDF is a statistical method used to assess the importance of a word to an article or article in a corpus. The importance of a word increases directly with the number of times it appears in a document, but decreases inversely with the frequency of its appearance in a corpus.

TF (Term Frequency) refers to word frequency, that is, the number of times a word appears in an article. However, there is a loophole in practical application, that is, the number of times a given word appears in a long article will be a little more. So we need to normalize the frequency, usually by dividing the frequency of a given word by the total number of words in the article.

IDF (Inverse Document Frequency), if the fewer documents contain the keyword x, it indicates that the keyword x has a good classification ability. The IDF of a keyword can be obtained by dividing the total number of articles by the number of articles containing the keyword, and then taking the logarithm of the result as seen in Figure 3.

In sklearn, TF-IDF needs import first: from sklearn˙feature_extraction.text import TfidfVectorizer˙Then create a new TfidfVectorizer instance and use fit_transform to transform title and context to number array. In this way, the plain text can be turned into a vector that can be used to train the model.

TF-IDF is simple and fast, the result is more in line with the actual situation. However, simply using "word frequency" to measure the importance of a word is not comprehensive enough, and sometimes important words may not appear many times; Can not reflect the position information of the word. Multiple document comparisons are required.

3.2.2 One Hot Encoding

Another feature extraction method that was employed is One-Hot Encoding (OHE). OHE assigns every unique entry to its category and was applied to the author column in our dataset. The rationale behind this approach is to recognise authors who appear frequently, ascertain the presence of a clear author, and gauge how this information impacts the reliability of the news. For instance, we desire our model to recognise an author who consistently appears in the dataset and has penned multiple reliable articles as likely trustworthy. Conversely, we aim for the model to consider an article lacking a clear author as potentially less reliable. Grasping how OHE operates is crucial since there are various ways to categorise input; an elucidative comparison is depicted in Figure 4, contrasting Label Encoding with One-Hot Encoding.

Unlike Label Encoding, which allocates labels ranging from 0 to N-1, where N represents the total number of categories, OHE adopts a binary approach wherein each new author is accorded a new column. This distinction is particularly pivotal as it prevents the model from misinterpreting the categorical data. Label Encoding could inadvertently introduce a notion of order among the categories, which may lead the model to derive inaccurate associations or trends. For instance, a label encoding of 1 for Author1 and 2 for Author2 may mislead the model into inferring a hierarchical or ordinal relationship between these authors, which is non-existent.

However, the use of OHE is somewhat questionable as the sample size is not nearly as large and consistent to be effective. The potency of OHE significantly hinges on having a substantial amount of data to adequately represent each category, in this case, each author. In our case, there are 4202 authors among 20800 entries, in which in average an author would only reappear around five times. With a limited or inconsistent sample size, the binary columns created through OHE may not provide a robust representation of the authors, potentially leading to a sparse matrix with many zeros and few ones. Additionally, a small sample size coupled with a vast number of unique authors could exacerbate the dimensionality issue associated with OHE, as each unique author translates to a new feature in the dataset. The resultant high-dimensional space may pose challenges for the machine learning model, including increased computational requirements and a higher susceptibility to overfitting, where the model learns to memorise the training data rather than generalising from it.

Regardless of OHE's clear disadvantage with the lack of data presently on the dataset we are using, we believe it is the only appropriate method to extract any useful information from the author column. Information such as names is challenging to translate into meaningful information that a machine learning model could leverage. Names, being inherently categorical and highly diverse, pose a significant challenge in deriving insightful information. They do not possess an ordinal quality nor a scalar representation that could be directly fed into machine learning algorithms. However, OHE provides a viable solution to this conundrum where it can efficiently translate authors' categorical data into a numerical format that machine learning algorithms can process. By doing so, OHE facilitates the conversion of the abstract representation of authorship into a tangible format that can be analysed and related to news reliability, which is the objective of our study. In conclusion, despite the known limitations, the advantages of employing OHE outweigh the disadvantages in our scenario. It offers a pathway to convert the challenging categorical data of author names into actionable insights, thus enhancing the potential of our machine-learning model to discern patterns related to news reliability.

4 Classification

4.1 Naive Bayes

Naives Bayes is a classification model that predicts the probability of a category given a set of features, which it then classifies the category based on the highest estimated probability.It is suitable for our machine learning as it presents some unique advantages such as being relatively scalable as it can handle large amounts of data quickly due to its low computational cost; however, it's simplicity means that it is unable to take in more details vs models such as Neural Networks. As such it serves as a good baseline comparison model against other ML models. In addition, the model is limited by it assuming the features are independent, which in the real world is usually not the case, thereby decreasing its accuracy. Using our data we produced a very accurate result of 98.9% accuracy. Below in figure 5 you can see the results of the confusion matrix from Figure 5.

However, when we use data sets outside of the training We see that the accuracy drops from almost 99 percent down to 61 percent. Even so, its 61% accuracy suggests that the model is better than guessing. As such, further models are used to examine whether the AI models can generate better results

4.2 K Nearest Neighbour

The k-nearest neighbor method (k-NN) is a classification and regression approach. The k-nearest neighbor method assumes that given a training dataset where instance categories are known, when classifying a new instance, it predicts based on the categories of its k nearest neighbors from the training instances using methods like majority voting. Therefore, k-nearest neighbor does not involve an explicit learning process. The choice of k, distance measurement, and classification decision rules are the three fundamental elements of k-nearest neighbor.

One reason for using the k-nearest neighbor is our project is a classify project, which is the k-nearest neighbor good at. And also for some other reason, k-NN can easily adapt to new categories. Here is text data and without having to retrain the entire model. This is useful for working with update text data. Last reason is it has robustness to outliers, k-nearest neighbor is not sensitive to outliers because it votes according to the nearest neighbor sample, and a single outlier does not have a significant effect on the classification.

In python sklearn library, k-nearest neighbor algorithm is already integrated. It is easily obtained by code: from sklearn.neighbors import KNeighborsClassifier. Then put all training data into the model's fit function. When fitting all training data, it can be called the predict function to predict.

When we use original data and set k = 5, It has really high accuracy, which is 0.9704326923. However when using data mining data from the internet, the accuracy becomes 54.98076923076923%, which is not good, just a little bit better than random guessing. When change k = 3 the accuracy is 54.90384615384616%, and when k = 2, it is 54.38461538461539%. However, it is not much different, it reaches the limit of knn model. And this model is the worst one compare the other three model. The reason is the knn not good at high dimension data, and when training this model knn need a single vector input, so the input x combine all feature to one vector which has 10000 dimensions. This is called peaking phenomenon, which means where after the algorithm reaches the optimal number of features, additional features increase the number of classification errors, especially when the sample size is smaller. That is why knn is the worst one, as seen in Figure 7.

4.3 Neural Network

Neural Network is the most successful model that has achieved the highest accuracy score among those our team has used. In essence, it is inspired by the human brain's structure and the ability to learn. The most crucial definition of neural networks is their structure. Neural networks are built using a tremendous amount of interconnected units, just like the neurons in our brains. These neurons are all connected, simulating the axons, which serve as a way of communication between these neurons. Another vital understanding is layers. Neural networks organise neurons into layers where inputs are put into the first layers and then processed through multiple layers to output what we need finally. It is important to know that the layers after the input layer is not visible and it is where input are processed into output. Understanding these two definitions, Figure 7 is a good representation of the architecture of Neural Network which we will try to replicate.

For Neural Networks, we have decided to use two different libraries to employ it. Specifically, we will use MLP from sklearn and Keras from Tensorflow. The main difference here is how structures were constructed and that, different from the remaining models, Keras uses Deep Learning. Both models delivered remarkable results, and Keras turned out to be the one that gives the best accuracy.

4.3.1 MLP - SKlearn

The multilayer perceptron (MLP) in "scikit-learn" is a type of shallow neural network. Essentially, it is a bridge between traditional machine learning models and more advanced deep learning models. Although it is equipped with multiple layers of neurons that can capture complex relationships in the data, its architecture is inherently simpler and shallower than Keras-based models. This design has both advantages and disadvantages. On the one hand, MLP provides faster training times and requires less computing power, making it suitable for lower complexity datasets. On the other hand, for highly complex datasets, such as those involving complex text patterns in news articles, the depth may not be sufficient to capture all potential nuances. In our project, although MLP has an accuracy of 62%, its accuracy is not that high compared to Keras. Nonetheless, MLP remains an important tool in machine learning, especially when computing resources are limited or the data is not complex.

We tried to use MLP in our project because of its ability to decipher multi-dimensional patterns in text data. News articles often contain complex context and need to be able to capture the underlying relationship between words and overall sentiment, which often embodies non-linear patterns, and MLP has the ability to capture non-linear relationships, making it a suitable choice. Although its architecture is simpler than a deep neural network, it is good at handling the complexity of text data, allowing it to discern subtle clues that may indicate the authenticity or falsity of an article. In addition, considering the risk of overfitting in high-dimensional text data, the relatively shallow structure of MLP plays a protective role.

To start the training process, we must ensure that our model is getting the correct information. First we need to clean the data, then we use the essence of the news article (title, author and text) as the main components for the model to learn. We can think of the MLP model as a multi-layer cake. The base layer of the cake, the input layer, obtains original information from the title, author and text of the news. On top of this we build a single hidden layer, similar to a layer of rich cream. This layer consists of 128 neurons and helps blend inputs to reveal complex patterns and relationships in the data. We first need to tell the model what fake news is and how to determine whether it is fake news. We feed the model training data to help it learn the difference between real news and fake news. After learning, we test it with a new data set to see if it can correctly identify whether they are true or false.

At the beginning of the project, we used a single-layer neural network and achieved an accuracy of around 95% on a familiar dataset, which is a high value for a complex news article. Then we tried to make it more accurate by adding hidden layers and increasing the number of neurons in each layer. We increased the original 1 hidden layer with 128 neurons per layer to 2 hidden layers with 256 neurons per layer and got the result. It's roughly the same as the original, still about 95% accurate. Since the results are of little significance, we finally chose a single hidden layer with relatively lower complexity and computational complexity. Applying it to a new data set, it achieved an accuracy of around 62%, which is second only to Keras among all the models we tried as we can see from Figure 8.

4.3.2 Dense Layer - Keras

In contrast to previous models, Keras is a deep-learning neural network model. This means it operates on multiple levels of abstraction to derive insights from data. The architecture of Keras facilitates the creation of more complex models by stacking multiple layers, each capable of capturing intricate patterns within the data. Unlike traditional machine learning models, which often operate on a single layer or a fixed number of layers, Keras, with its deep learning capability, can handle a higher number of layers, which empowers it to model complex, non-linear relationships more effectively. Moreover, it operates on top of TensorFlow, one of the fastest computational frameworks, making training the model extremely robust.

In order to train the model, layer construction is needed after the train/test split. To begin, the findings from feature extraction are channelled into the architecture of the neural Network through the input layer corresponding to title, author, and text. For each of these input , we have constructed two hidden layers with 128 and 64 neurons, respectively.

It can be very technical and complex to understand the reasonings behind these choices. However, the main objective is always to find balance, not to under or overtrain the model. Imagine we are trying to bake a simple cake, but we bring in a team of master chefs and a whole warehouse of ingredients. It could lead to too many ideas, too much complexity, and a huge mess, while a simple setup could have done the job perfectly. Similarly, having constructed way too many hidden layers or neurons can make the model more complex than it needs to be. It might start seeing patterns that are not there or are entirely irrelevant, making it less reliable for new, unseen data. In simple terms, this is called **overfitting** a model. Allocating such complexity also requires more computational power and time to train. On the other hand, imagine we are attempting to cook a feast for over 100 people. However, we only have a toaster, some rusted potatoes, and some vegetables. For sure we will not be able to make enough variety of food or in the quantity needed. Likewise, having too few neurons or hidden layers might make the model too simple to the point where it misses out on understanding the essential patterns. This is called **underfitting**.

Having understood the balance needed in setting up the neurons and hidden layers, it is crucial to study another core aspect of NN's architecture, that is, the **activation functions**. For a simple explanation, we will be using the scenario of a gameshow. In this context, activation functions are like the judges, determining which contestants (or, in our case, information) get to move forward to the next round and which gets eliminated. Now, envision a scenario where each piece of information from the text, title, and author is like a contestant in a game show, vying for a spot in the next round. They first go through a series of tests (layers) where they are tweaked and refined. This is where the **ReLU (Rectified Linear Unit) activation** comes into play. ReLU is like a lenient judge, allowing all positive performances to go through unaltered but stopping any negative performance in its tracks by setting it to zero. This ensures that only the useful and positive information moves forward in the game.

Once our contestants (pieces of information) have gone through these initial rounds, it is time to bring them together on a common stage for the grand finale. This common stage is created through a process called **concatenation**, where all the refined information from the text, title, and author are brought together, ready to be judged as a whole. As they enter the final stage of the game, they encounter one last judge - the **sigmoid activation** function. Unlike ReLU, the sigmoid judge is more nuanced and calculates a score between 0 and 1 for each contestant, signifying the probability of them belonging to a particular category - in our case, the likelihood of the news being reliable.

Lastly is the training and result. There are two essential terms that we need to understand here: **epochs** and **batch size**. Epochs refer to the number of times the model goes through the training data, each time learning and refining its understanding. For ours, we found that the best number is 25, which is when the model achieves its peak accuracy. Batch size refers to the number of data entries we predict at a time. While having each entry train at once can improve performance, however, this is a very computationally expensive task in which using batch size can help the training to be much more manageable and efficient. As for results, keras achieved an accuracy score of 99.85%, the highest among the four models. Our test dataset, which consists of 4160 articles, was predicted, and the results are shown in Figure 9.

Figure 9 shows us four critical pieces of information:

- **True Positive (bottom right: 2026)**: This number represents the articles that were *actually* real and were *correctly predicted* by the model as real.
- **True Negative (top left: 2128)**: This number represents the articles that were fake and were *correctly predicted* by the model as fake.
- **False Positive (top right: 4)**: This number represents the articles that were fake but were *incorrectly predicted* by the model as real. In other words, the model mistakenly thought these fake articles were real.
- **False Negative (bottom left: 2)**: This number represents the articles that were *actually* real but were *incorrectly predicted* by the model as fake. The model mistakenly thought these real articles were fake.

Overall, Keras Neural Network was a solid attempt to solve this problem. However, we also realised the shortcoming of this solution is the amount of information. On a familiar dataset, all of our models performed exceptionally well, scoring more than 90%. However, they all dipped when introduced to a new dataset and for Keras, the accuracy score was reduced to almost 68%. 68% is still the best score out of the four models, and it is still a decent score given that it is better than guessing. However, specific alterations must be implemented to make a better detection tool.

5 Simulation Results

5.1 Key Findings and Significance

In our finding we find that even though the machine learning models were able to do fairly well on the training dataset, it wasn't able to predict the results accurately. This indicates a couple of thing, the models are likely over-fitting to the training data set. This could be due to the fact that the dataset was cleaned too "well", causing the models to develop bias towards specific behaviors. This can happen easily as the feature extraction has an heavy emphasis on frequency of words, and the training dataset could contain fake articles that favored specific words. As a result, when the ML models were used outside this scope, it performed badly as other fake articles may not use the same words. In addition, we also found it interesting that some models performed better. For instance, Naives Bayes model performed very well, where as we see a dip in accuracy for KNN and MLP. With the Dense Layer from Keras performing the best as the figure 9 shows. We believe that the KNN performed worse due to it suffering from the curse of dimensionality, causing it to classify the articles incorrectly. KNN and MLP are also more succestiple to noisy data, as some of the articles can contain irrelevant features to throw them off; however, higher accuracy that comes from the Dense Layer indicates that using more complex frameworks that seek to mitigate these downsides can increase the accuracy.

5.2 Commercialization and Translation Opportunities

There are tremendous opportunities that can happen based on the findings. The findings do not yet support the creation of a tool for a general user to classify whether an article they are reading is fake or not as the percentages are too low. However, the fact that there are differences in accuracy for ML models indicates that researcher's should focus on models that would suit fake new detection. In addition, this paper does not explore the impact that different feature extractions can have on the accuracy of the model, such as n-gram, sentimental analysis, and other ones. As such more research can be held in this area

This research paper believes that even though there is potential for software engineers to develop a tool for for social media companies, such as Facebook, and news articles sites to develop a tool that detects fake news as a prevention tool instead. This is due to the fact fake news can spread in many different forms such as images, voice, and videos. An additional reason for co-operation to work together to develop a model is due to the fact that these models are limited by it's inability to crosscheck between different news-sources. One of the key ways to detect fake news is by seeing whether other reputable news sites are reporting the same thing, the paper's models can only detect via patterns deduced from feature extracted data.

6 Conclusions

As such the paper finds that there are limited commercialization for the currently developed ML model, as it is too inaccurate to be used in a real world scenario; however, its "better than guessing" accuracy as well as indication that there are better ml models that suit the classification problem that is fake news detection points to tremendous opportunities to expand on this research. In addition, the paper implores that future ML models to develop techniques that can cross-reference other news sources, as the current model can only examine the source text. Other large organizations can also use the paper as indication to develop their own ML models, as they have access to more resources, such as their own new articles, as well as the ability to collaborate with other reputable news sources in order to train their own models. As such, the paper illuminates that ML models can indeed, with some accuracy, predict whether a news article is real or fake.

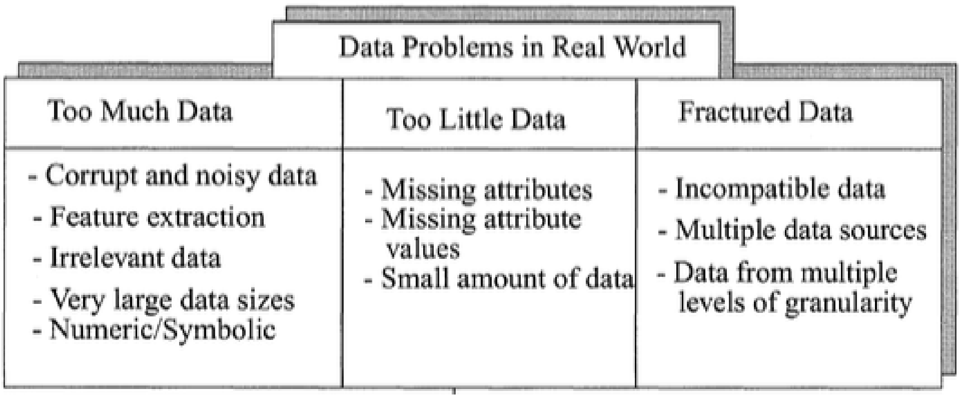
References

[1] A. FAMILI, W. SHEN, R. WEBER, and E. SIMOUDIS, "Data preprocessing and intelligent data analysis," *Intelligent Data Analysis*, vol. 1, pp. 3–23, 1997.

[2] r. , "What is one hot encoding? why and when do you have to use it? — hacker noon," 08 2017.

[3] "Neural network architectures," 07 2019.

7 Appendix



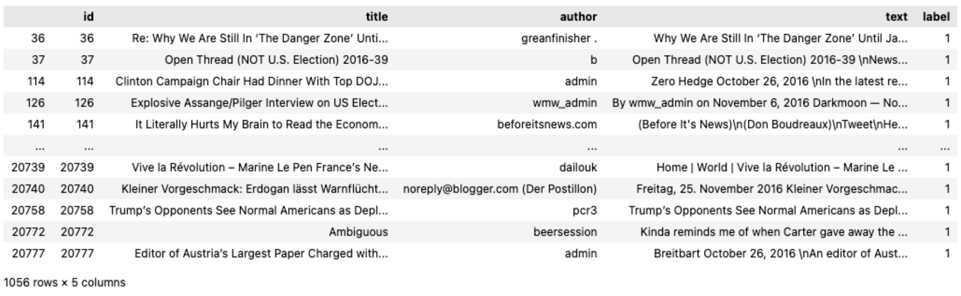
Data Problems in Real World

Too Much Data	Too Little Data	Fractured Data
- Corrupt and noisy data - Feature extraction - Irrelevant data - Very large data sizes - Numeric/Symbolic	- Missing attributes - Missing attribute values - Small amount of data	- Incompatible data - Multiple data sources - Data from multiple levels of granularity

Figure 1: Data Problems in the Real World [1]

Field	Value
id	0
title	558
author	1957
text	39
label	0

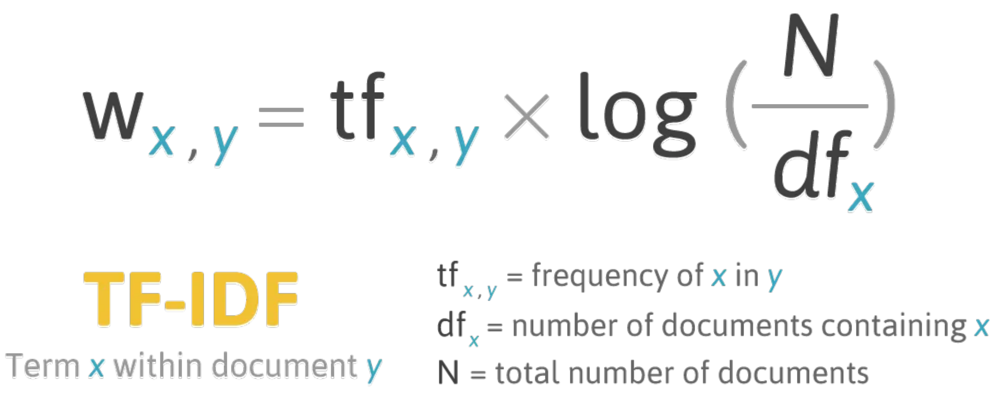
Table 1: Null Values in Dataset



	id	title	author	text	label
36	36	Re: Why We Are Still In 'The Danger Zone' Unti...	greanfinisher .	Why We Are Still In 'The Danger Zone' Until Ja...	1
37	37	Open Thread (NOT U.S. Election) 2016-39	b	Open Thread (NOT U.S. Election) 2016-39 \nNews...	1
114	114	Clinton Campaign Chair Had Dinner With Top DOJ...	admin	Zero Hedge October 26, 2016 \nIn the latest re...	1
126	126	Explosive Assange/Pilger Interview on US Elect...	wmw_admin	By wmw_admin on November 6, 2016 Darkmoon — No...	1
141	141	It Literally Hurts My Brain to Read the Econom...	beforeitsnews.com	(Before It's News)\n(Don Boudreaux)\nTweet\nHe...	1
...	...	...	...	...	...
20739	20739	Vive la Révolution – Marine Le Pen France's Ne...	dailouk	Home | World | Vive la Révolution – Marine Le ...	1
20740	20740	Kleiner Vorgeschmack: Erdogan lässt Warnflücht...	noreply@blogger.com (Der Postillon)	Freitag, 25. November 2016 Kleiner Vorgeschmac...	1
20758	20758	Trump's Opponents See Normal Americans as Depl...	pcr3	Trump's Opponents See Normal Americans as Depl...	1
20772	20772	Ambiguous	beersession	Kinda reminds me of when Carter gave away the ...	1
20777	20777	Editor of Austria's Largest Paper Charged with...	admin	Breitbart October 26, 2016 \nAn editor of Aust...	1

1056 rows × 5 columns

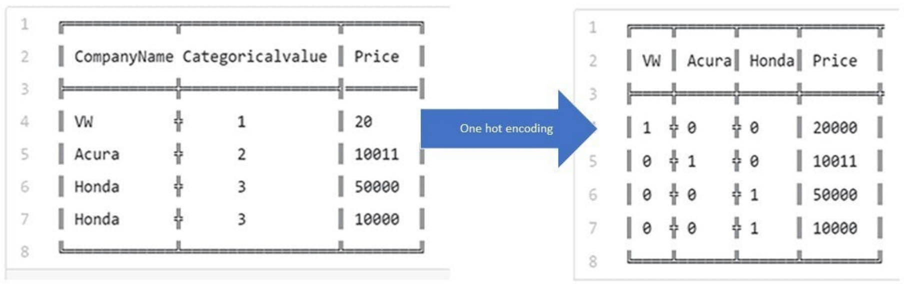
Figure 2: Ambiguous Authors



$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

Figure 3: TF-IDF



CompanyName	Categoricalvalue	Price
VW	1	20
Acura	2	10011
Honda	3	50000
Honda	3	10000

One hot encoding

VW	Acura	Honda	Price
1	0	0	20000
0	1	0	10011
0	0	1	50000
0	0	1	10000

Figure 4: Label Encoding vs. OHE [2]

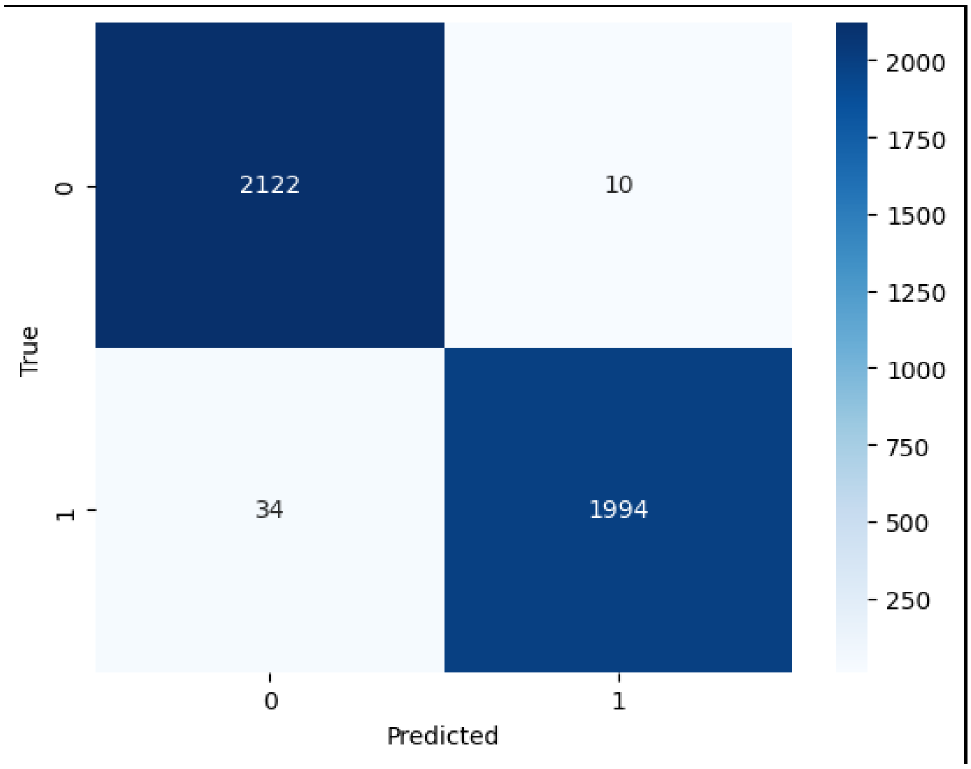


Figure 5: Naive Bayes Confusion Matrix [3]

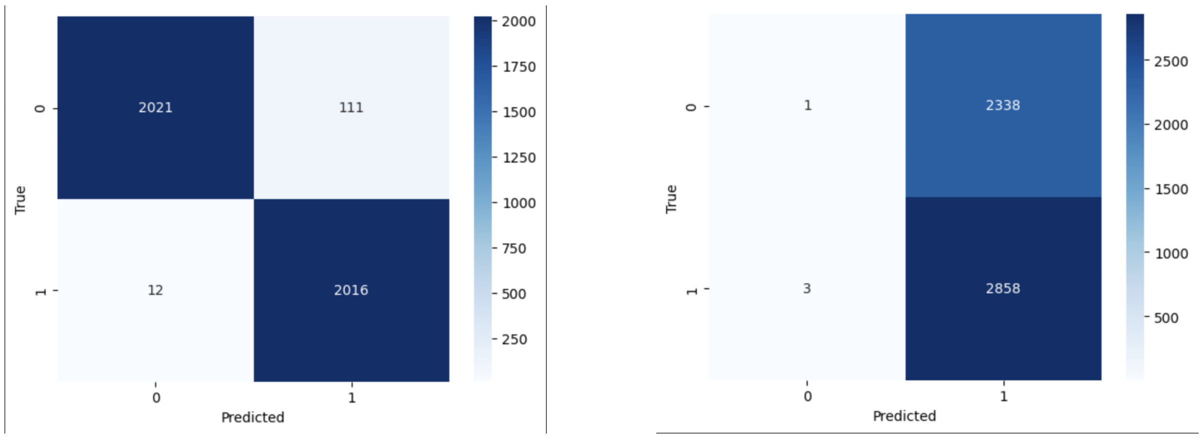


Figure 6: Confusion Matrix With New and Original Data Of KNN

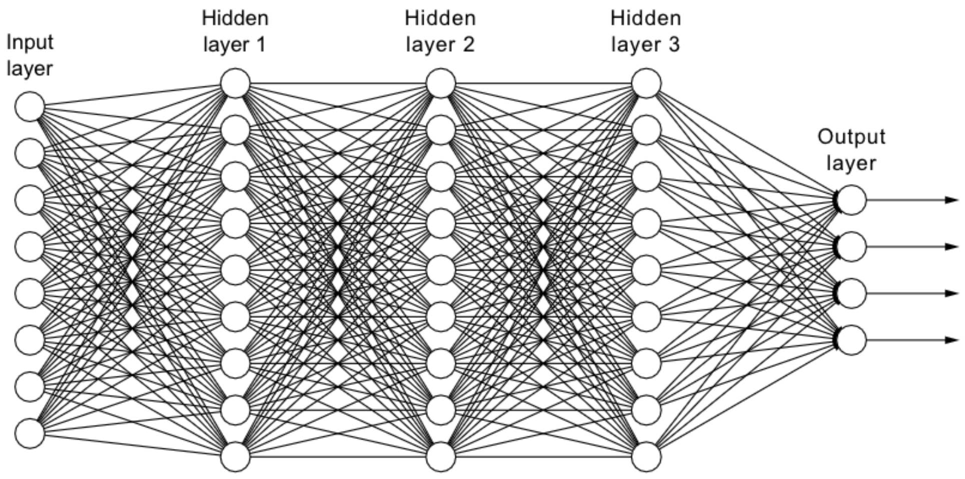


Figure 7: Neural Network Structure Example [3]

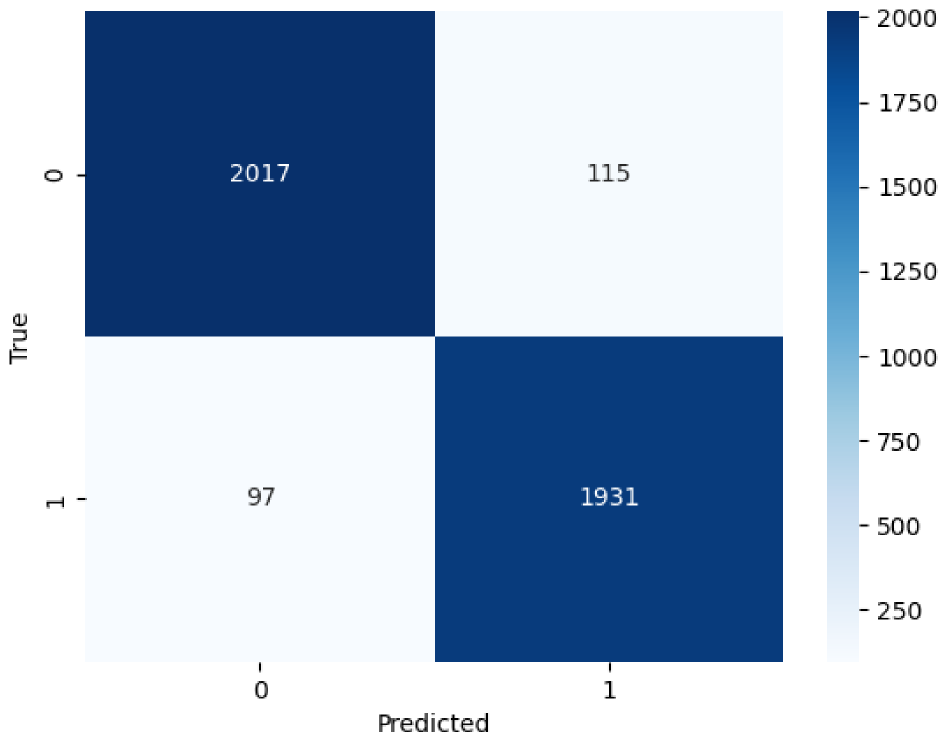


Figure 8: MLP's Confusion Matrix

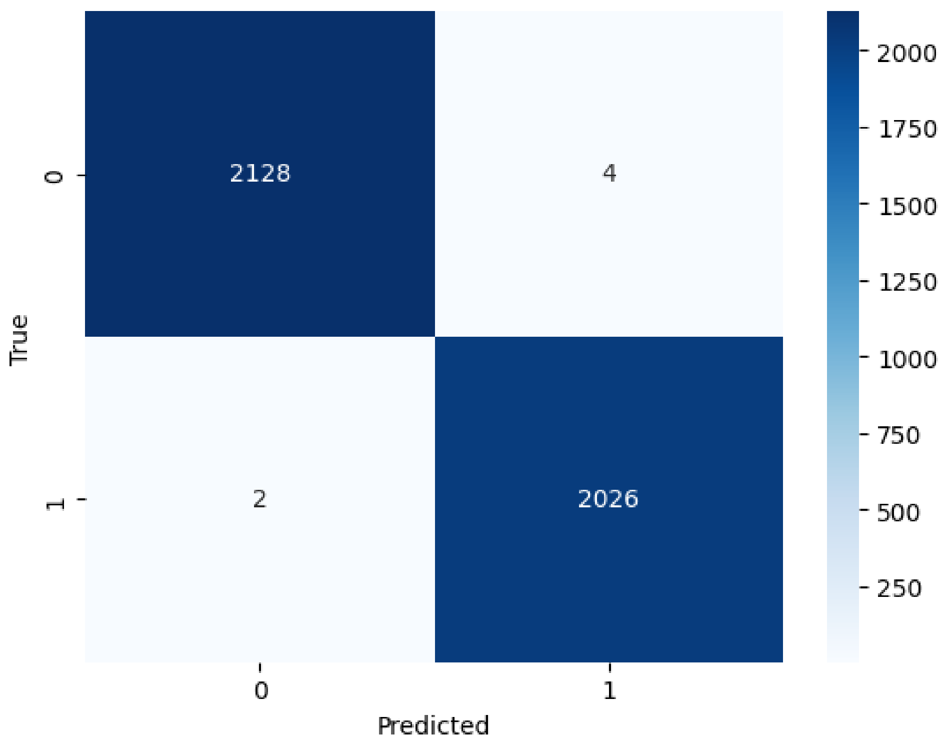


Figure 9: Dense's Confusion Matrix

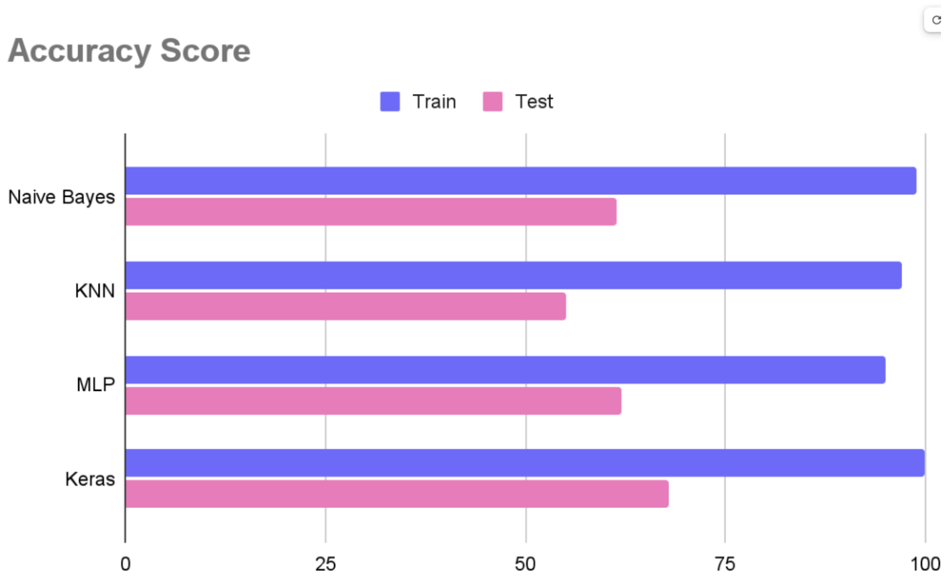


Figure 10: Results comparing 4 different models