Fake News Classification using Machine Learning algorithms.

Filippos Rogdakis, Emmanouil Smpokos, Orestis Zoumpoulakis

Abstract

In this paper, we introduce a range of machine learning models designed to accurately classify news articles containing fake news. Our primary objective is to create an algorithm capable of scrutinizing the content of the article to determine its authenticity of the article. To achieve this, we applied three machine learning models: Logistic Regression, Support Vector Machine (SVM) and Random Forest. These models were chosen based on our research on distinct research groups that had similar objectives (such as [1], [2]). Ultimately, we will compare our results with the aforementioned studies.

Keywords: Fake News, Machine Learning Models, Classification, SVM, Random Forest, Logistic Regression

1 Introduction

The spread of fake news nowadays has become easier due to the rise of digital platforms, social media. Some individuals may spread fake news for personal or financial gain. Clickbait headlines and sensational content can be used to attract more clicks and views, ignoring the main purpose of media which would be to inform the reader. Furthermore, many readers lack the time or the knowledge to 'filter' authentic from fake news. But at the same time, there has been a rapid and expansive development in the field of artificial intelligence, especially in machine learning and data analysis. This progression has offered us limitless options to what we can do with the right tools and knowledge. Having this in mind, we used certain scripts to analyze our data in order to use the various ML models, elaborated in the subsequent sections, to determine the authenticity of the news articles within our data.

2 Methodology

2.1 Data Selection

Our data are derived from Kaggle.com, a site with various machine learning competitions and projects. The dataset selected contains the title, text and author of each news article as well as an id and a label (0 or 1, where 0 is Fake and 1 is Real).Below we show the sum of instances in each class in the form of a bar chart.

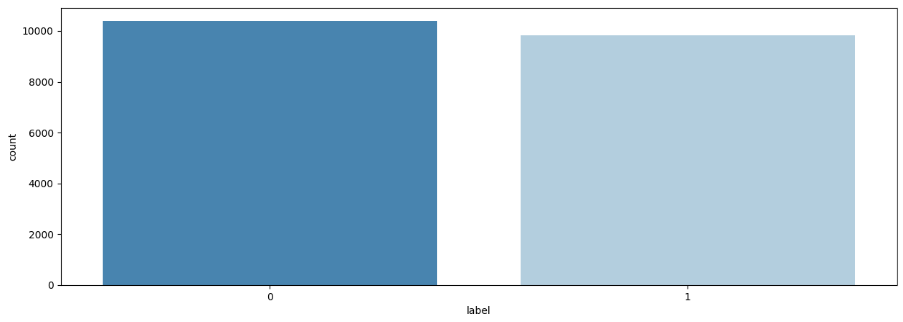


Figure 1: Caption

2.2 Preprocessing

Since our classification is text-based, preprocessing of our data is a crucial step of our analysis.Our preprocessing consists of:

2.2.1 "Cleaning" of text.

We start our preprocessing by cleaning our text and in order to do so we have to first fill all null values in our data-set,then remove all the numbers of the text (as they don't contribute significantly in determining the credibility of news) and lastly transform all letter to lowercase to have a more consistent data-set.

2.2.2 Sentence Segmentation and Tokenization

During this process,punctuation marks such as exclamation(!), full stop(.) etc. are used to brake the text into sentences. Then, each sentence is broken up into words (tokens) by finding markers/delimeters ('Tab','space',comma etc.) inside the sentence.

2.2.3 Stopwords Removal

To reduce the size of our dataset and make our classification more efficient, we also have to remove stopwords from text. Stopwords are words that are found very frequently inside text and do not contribute to the meaning of it.The following words('a','an','the','of','in','you') and more, are some examples of what was removed from our text.

2.2.4 POS tagging

Part-Of-Speech tagging is an important step, in our preprocessing, that involves assigning a grammatical category (noun, verb, adjective) to each word in order to analyze and understand the syntactic structure of each sentence.

2.2.5 Words Lemmatization and Stemming

Word Lemmatization and Stemming are two different methods that can be used to reduce words to their lemma (root word).The Lemmatization algorithm uses the previously mentioned POS tagging, takes into consideration the context of the sentence and then reduces each word to its lemma. Stemming on the other hand, is a more 'brute force' approach that removes suffixes or prefixes from every word. By performing the above, all forms of a word are 'normalized' to their root. Our classification algorithm can then work more efficient on identifying the words that are more often used in fake news.

3 Feature Extraction

Feature extraction is the process of transforming the raw data into numerical values. Text cannot be understood directly by ML algorithms and must be encoded in a format that ML algorithms can use. The numeric vectors can be understood by ML algorithms. Our dataset is composed of text so in order for our models to be able to be applied, feature extraction must be performed. It also helps to reduce the amount of redundant data from our data set. We used the Term Frequency-Inverted Document Frequency (TF-IDF) method which is a statistical metric used to measure the importance of a term in a data set. Its importance increases with the number of times a word appears in the document, but this is counteracted by its frequency in the corpus. With this vectorizer we weight down words such as 'the' and 'then' that don't contribute to the document.

4 Classification

4.1 Logistic Regression

We know that regression algorithms can be used as classifiers in some cases.Since our classification is binary, Logistic Regression can be used as a powerful classifier. The Logistic Regression algorithm tries to calculate the probability that an instance of our data belongs to a certain class. It does so by fitting the data with a sigmoid function as seen below:

$$\mathbf{y} = \frac{1}{1 + e^{-\mathbf{wx}}}$$

where $\mathbf{x}$ and $\mathbf{w}$ are vectors or matrices of the instances and their weights respectively.

Training our algorithm, in this case, means finding the optimal weight vector $\mathbf{w}$ which is done by minimizing the conditional log likelihood using Gradient Decent .

Conditional log likelihood can be expressed as:

$$l(\mathbf{w}) := -\sum_i logP(y_i|x_i, \mathbf{w})$$

And in the form of a cost function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log(\hat{p}^{(i)}) - (1 - y^{(i)})log(1 - \hat{p}^{(i)})\right]$$

Which (as seen in the equation above) is just the log of the conditional likelihood $P(y_i|x_i, \mathbf{w})$: The probability that the observed class y is was a correct assumption based on the conditions x_i and $\mathbf{w}$.

So $\mathbf{w}$ can be calculated as:

$$\mathbf{w} = \text{argmin}_{\mathbf{w}}\text{l}(\mathbf{w})$$

Using the formulation of Gradient Decent, $\mathbf{w}' = \mathbf{w} - \eta\nabla l(\mathbf{w})$,until l($\mathbf{w}$) is minimum.

Our efforts do not stop there though. We have to make certain that our model does not over-fit to our train data and one way to do that is through a process called regularization. For this classifier we tested to ways of regularization,which are briefly discussed below.

4.1.1 L1 Regularization (Lasso Regularization)

The way this method (or any method of regularization) works is quite simple. You add one term to the cost function and now the algorithm has to ,not only fit the data, but it to keep the model weights as small as possible.

The term added for L1 is the following:

$$a\sum_i |w_i|,$$

Where a is the regularization constant and w$_i$ is an instance of the weights.

4.1.2 L2 Regularization (Ridge Regularization)

The L2 method works similarly with the L1 method but the term added to the cost function now looks like this:

$$\frac{a}{2}\sum_i (w_i)^2,$$

After adding the regularization term, we continue the process ,as described above, by minimizing the cost function using Gradient Decent.

4.2 SVM Classifier

The Support Vector Machine(SVM) model tries to fit the largest possible 'margin', or gap, between the different classes of data. The points touching the margin are called support vectors, thus the name. The classification hyperplane and the cost function can be written as:

$$h(x) = \mathbf{w}^T x + b, \quad L(w, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$$

such that $y_i(wx_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0 \ \forall x_i$, where w is a vector perpendicular to the hyperplane, ξ_i represents the loss, it's $0 < \xi_i \leq 1$ for a point on the correct side of the hyperplane and $\xi_i > 1$ for the misclassified points. The hyperparameter C determines the trade-off between maximizing the margin and minimizing the misclassification error. A smaller C creates a wider margin but allows more margin violations, while a larger C creates a narrower margin and allows fewer violations.

If the data is not linearly separable we use the kernel trick. We move our data to higher dimensional spaces(even ∞) so now instrad of minimazing the cost function we need to minimize the following:

$$\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i^T x_j$$

s.t. $C \geq \alpha_i \geq 0 \ \forall x_i$ and . To address this problem we define a function K and we assume that $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ for some feature space $\Phi(x)$. We can choose any function K as well as it's well behaved. In SVM the choices for a kernel are linear, sigmoid, polynomial, and a gaussian.

It's important to scale the data before fitting because the algorithm calculates distances between the points, so points on larger scales can dominate this process, which can lead to sub-optimal performance of the SVM on the unscaled features.

What we did was scale the data using the with the StandardScaler method, then we fitted on the train the SVM classifier and predicted with the default hyperparameters just to get an idea of the accuracy of the model. Then, for hyperparameter tuning we run a GridSearchCV to find the best C value and the best kernel. We found that the best hyperparameters for the SVM model are C=20 and kernel=linear. We didn't allow the C value to take very large values in order to not overfit and to let our model to be general. For those parameters we then present the classification report and the confusion matrix.

4.3 Random Forest Classifier

Random Forest is an averaging algorithm based on randomized decision trees. Each tree is trained using the bagging method. Specifically, given a training set $X = x_1, \ldots, x_n$ with the corresponding responses $Y = y_1, \ldots, y_n$, the process of bagging involves iteratively choosing random samples with replacement from the training set and creating trees based on the samples (B times):
For b=1,...,B it samples, with replacement, n training instances from X,Y, denote them as X_b, Y_b, and then train a classification or regression tree f_b on X_b, Y_b. Following the training process, prediction for unseen samples x' can be generated by averaging the predictions obtained from each individual regression tree applied to x' (in case of the classification trees, it takes the plurality vote):

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

This method, coupled with a modified tree learning algorithm, introduces diversity in the training that process, which helps mitigate overfitting, a common issue with individual decision trees. In addition, during the construction of each tree, randomness is introduced by considering only a random subset of features when deciding the best split at each node.
Having done the necessary pre-processing, we created the Random Forest model. The "n_jobs = -1" parameter allows algorithm to utilize all the available CPU Cores of the system for paraller processing, which noticeably speeds up the training process. In some cases "n_jobs = 1" might do a better job depending on the system, so both must be tried in a new system. The "random_state = 42" parameter ensured reproducibility by fixing the random seed.

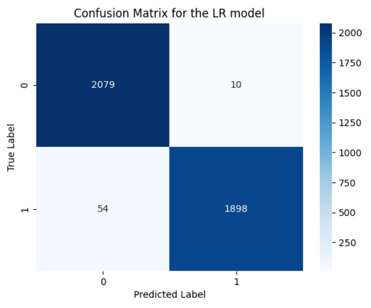
By running the model with the default hyperparameters we ensured that the model was running satisfactorily. Then, again for the hyperparameter tuning as we did on the previous methods, we run a GridSearchCV to find the best "n_estimators" , "max_depth" , "max_features" and "criterion". We found that the best hyperparameters for the RF model are the following:

- n_estimators = 100
- max_depth = None
- max_features = 'sqrt'
- 'criterion' = 'gini'

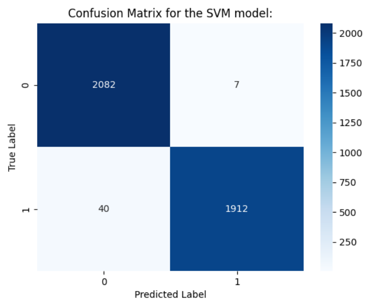
The parameter "n_estimators" represents the number of trees in the RF model, the "max_depth" parameter controls the maximum depth of every single tree and specifically "None" means that having no restriction in the depth yields better results. In addition, "max_features" determines the maximum number of features considered for splitting a node and in our case "sqrt" considers the square root of the total number of features. Finally, the "criterion" parameter measures the quality of a split and 'gini' refers to the Gini impurity, which is commonly used in decision tree-based algorithms for classification tasks.
As mentioned above, the score that determined the best parameters is accuracy because we want to maximize the correct predictions. For those parameters we then present the classification report and the confusion matrix

5 Results

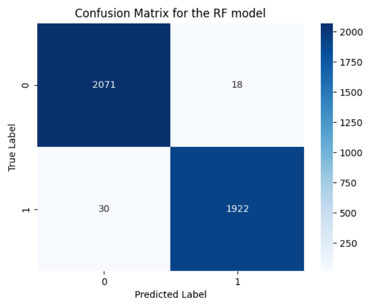
Below we present the Confusion Matrix and the classification report of each classification after hyperparameter tuning with GridSearchCV.



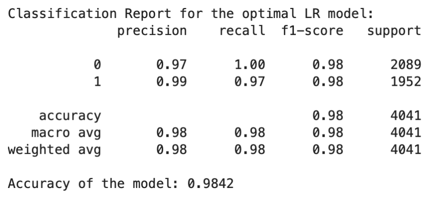
(a)



(b)



(c)



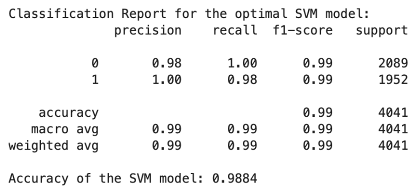
```
Classification Report for the optimal LR model:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      2089
           1       0.99      0.97      0.98      1952

    accuracy                           0.98      4041
   macro avg       0.98      0.98      0.98      4041
weighted avg       0.98      0.98      0.98      4041

Accuracy of the model: 0.9842
```

(a) LR model with hyperparameters: C=5, solver='liblinear', penalty='l2'



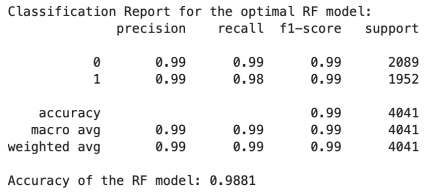
```
Classification Report for the optimal SVM model:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      2089
           1       1.00      0.98      0.99      1952

    accuracy                           0.99      4041
   macro avg       0.99      0.99      0.99      4041
weighted avg       0.99      0.99      0.99      4041

Accuracy of the SVM model: 0.9884
```

(b) SVM model with hyperparameters: C=20,kernel='linear'



```
Classification Report for the optimal RF model:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2089
           1       0.99      0.98      0.99      1952

    accuracy                           0.99      4041
   macro avg       0.99      0.99      0.99      4041
weighted avg       0.99      0.99      0.99      4041

Accuracy of the RF model: 0.9881
```

(c) RF model with hyperparameters: criterion:'gini',max depth: None,max features: 'sqrt',number of estimators: 100

6 Discussion

By comparing our work with the work of [1] and [2] we realise that our algorithms and especially Logistic Regression did a much better job. Although this might be good news for us we didn't accept it before taking into consideration that our model may over-fit our data. We did investigate on that and found no apparent case of over-fitting.

This may suggest that, in the context of the specific fake news dataset used, logistic regression's simplicity is competitive with the more complex random forest and SVM models but we intend to investigate further on other datasets.

7 Conclusion

In summary, the LR model has an accuracy score of 98.52%, the SVM 98.84% and the RF 98.81%. As we can see our models do a good job in detecting if an article or document is fake. So although the differences in their accuracy scores are insignificant, we would have to choose the SVM model to be the best due to it's better accuracy.

The purpose of our work is to create a ML model that is able to detect fake and authentic news. Inspired by that we can aim for future development of applications and browser extensions that will identify fake news and will inform the reader about the credibility of any article. We even intend that it can be implemented in a software that checks even the browser. We really hope that our research will contribute in a future where the amount of fake news in the media and the misinformation spread rate will be much lower.

References

[1] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. *Lecture Notes in Computer Science*, page 127–138, 2017.

[2] Avinash Bharadwaj, Brinda Ashar, Parshva Barbhaya, Ruchir Bhatia, and Zaheed Shamsuddin Shaikh. Source based fake news classification using machine learning. 2020.

[3] Muhammad Fayaz, Atif Khan, Muhammad Bilal, and Sana Ullah Khan. Machine learning for fake news classification with optimal feature selection. *Soft Computing*, 26(16):7763–7771, 2022.

[4] Aurélien Géron. *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools and techniques to build Intelligent Systems.* O'Reilly, 2020.

[5] Samrat Sinha. Fake news detection, Aug 2020.