An Empirical Comparison of Unsupervised and Supervised Classification for Fake News Detection

[redacted] Michael Baluja

Abstract—**A major challenge for media outlets, especially social media, is the spreading of misinformation. To avoid negative impacts on society, it is in our best interest to detect false information on social media platforms. Unfortunately, the amount of fake news is proliferating, and we need to find new methods to increase detection accuracy. This paper utilizes unsupervised and supervised models on real and fake news datasets to compare classification performance. Here we use K-means clustering and Support Vector Machines (SVM). We perform an empirical analysis on the performance of the two models to see which method yields the highest accuracy.**

I. Introduction

Through dissemination, fake news feeds on sensationalism, hoaxes, conspiracies, false rumors, and scandals. While this is not a new phenomenon on the internet, the extent to which it proliferates through social media increases its negative impact on society. Harmful intent is often debated; however, incentives such as monetary, social, and political benefits are often drivers for spreading fake news [1]. The spread of fake news is a crucial problem due to its unexpected consequences and potential for large-scale turmoil triggering [2].

News sites and social media are used by millions of people every day. In addition, people around the world demonstrate their feelings and opinions based on these sites every second. Consequently, information is distributed very quickly, and its impacts on social networks are critical since it can be reinforced and affect an incredulous number of people in a matter of minutes [3]. Therefore, detecting fake news has become an increasingly popular new research topic in recent years.

One such medium is Reddit's r/news and r/worldnews subreddits, where four hundred posts and twenty thousand comments are posted on a given day. Reddit has made efforts to combat fake news through manual moderation and automated systems. However, manual moderation does not scale well, and moderators themselves may be susceptible to misinformation and biases. The automoderator is a useful system customizable to a given subreddit's needs but lacks specificity. A classifier capable of distinguishing between real and fake news would be invaluable for these and many other social media platforms.

Supervised classifiers are an attractive group of algorithms for class-based distinction tasks in instances where class labels are known before training. However, for tasks such as fake news detection, this is oftentimes unattainable, as the authors behind many "fake" news platforms do not wish to advertise the illegitimacy of their articles. For tasks such as recognizing fake news "in the wild," it is, therefore, necessary to use some type of unsupervised algorithm to aid in our classification efforts. To understand how well the unsupervised classifier is performing, we utilize labeled training data. However, we note that this is different from simply using such labeled data to train a supervised classifier because we only employ the labels for measuring performance instead of using them during training.

The project's main objective is to improve differentiation between fake and genuine news using supervised classifiers SVM and RF and unsupervised classifiers K-means and MoG. Clustering refers to grouping similar data points together, based on their features, such that each cluster holds the most similar points. For example, we want to see how we can cluster the real and fake news using the sentiment features and vectorized text with and without dimensionality reduction techniques on the news articles.

II. Literature Review

For the classification of real and fake news, there exist two categories of significant research: the first category is related to approaches at a conceptual level, including fake news distinction concerning serious lies (the news is pertaining to unreal events or information), tricks (such as, deliberately providing wrong information), and comics (for example, funny news that imitates real news with strange contents) [4], while the second category is related to approaches at a linguistic level utilizing practical techniques to compare the contents of real and fake news [5].

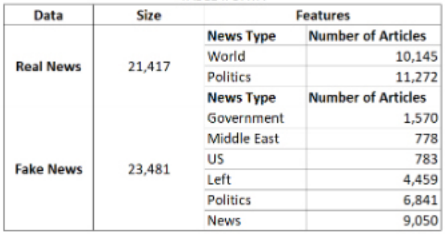
The linguistic approach uses specific linguistic behaviors such as markings, vocabulary, or labeling that are considered unintentional and beyond the scope of an author's attention. Through appropriate evaluation with linguistic techniques, promising results are revealed in the detection of fake news.

Majbouri et al. [6] propose a method of dimensionality reduction to extract only essential features of high dimensionality datasets containing real and fake news articles. The steps in their proposed method are computing similarity between primary features in the fake news dataset, clustering features by similarity, and detecting fake news using an SVM classifier. The proposed method was evaluated by comparing performance with Decision Tree and Naïve Bayes methods and found that classification accuracy improved because of redundant feature elimination and dimension reduction.

III. Data

In this project, to classify news as real or fake, we use the ISOT Fake News Dataset [7][8] gathered from the Kaggle website. The data consists of two CSV files, "True.csv" and "Fake.csv." Each article includes the following information: article, text, type, and the date of publication (see Table I). The data was cleaned and processed as explained in the following methods section of this paper.

TABLE I. DATA



Data	Size	Features	
Real News	21,417	News Type	Number of Articles
		World	10,145
		Politics	11,272
Fake News	23,481	News Type	Number of Articles
		Government	1,570
		Middle East	778
		US	783
		Left	4,459
		Politics	6,841
		News	9,050

IV. Methods

First, we clean our data in preparation for the supervised and unsupervised algorithms. Then, once cleaned, we introduce sentiment analysis, and term frequency-inverse document frequency (TF-IDF) a form of feature generation to enhance classification performance on the classifiers.

A. Data Cleaning

Before pre-processing our data for analysis by performing TF-IDF and sentiment analysis with the Natural Language Toolkit (NLTK) [9] to ensure it was ready for analysis under the supervised and unsupervised methods, we employed data cleaning techniques. For simplicity, we elected to analyze only the titles of the news articles contained in our datasets. Therefore, before we could perform sentiment analysis on the data, it was necessary to clean the text by reversing contractions, and unfiltering profanity using dictionary mappings, removing username handles (e.g., @name), separating words contained in hashtags by removing pound symbols (e.g., #HashTag becomes Hash Tag), as well as removing English stop words, punctuation, and website links.

B. Sentiment Analysis

Once we had cleaned the data, we implemented sentiment analysis and TF-IDF on the titles. Sentiment analysis is a text analysis tool that detects the polarity (e.g., a positive or negative sentiment) within the text. To detect polarity in the titles of our data, we utilized NLTK's VADER lexicon, a sentiment analysis model that is sensitive to both polarity and sensitivity of emotion [9]. The sentiment score is obtained by taking the sum of the intensity of each word in the text. Sentiment analysis provided us with additional features to use for classification performance.

TF-IDF is a numerical statistic reflecting the importance of a word for a document in a corpus [10]. We apply TF-IDF to the titles based on a constrained vocabulary of the top 2000 most frequent words, which is reduced from the original vocabulary size of 20,874. These TF-IDF features become the data we wish to train on, and the 0/1 label for fake and real news becomes the labels that enable performance measurement.

TF-IDF provided 2000 new features for the models to train on; however, this is an exuberant amount of features. Principal component analysis (PCA) is a well-known technique for applications such as dimensionality reduction [11]. We, therefore, implemented PCA on the TF-IDF features for some of the model training.

C. Unsupervised Learning Methods

The goal of K-means clustering is to partition the dataset into K number of clusters after several iterations. Performance is dependent on convergence to optimal local points (centers). K-means begins with randomly selected locations as initial cluster centers. During each iteration, cluster centers are updated based on the data points nearest to them and continue until improvement halts. We implement K-means manually following Bishop implementation [11] as in the second homework assignment and with Scikit-Learn [12] implementation.

For K-means, we introduce a 1-of-K coding scheme where we have a corresponding set of binary indicator variables $r_{nk} \epsilon \{0,1\}$ where $k = 1, ..., K$ describing which of the K clusters a data point x_n is assigned to cluster k then $r_{nk} = 1$ and $r_{nj} = 0$ for $j \neq k$. We define an objective function as,

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

which represents the sum of squares of the distances of each data point to its assigned vector μ_k [11].

Expectation step: For each data point x_n, we assign to the closest cluster; in this step, we minimize J with respect to r_{nk} keeping the μ_k fixed. We assign the n^{th} data point to the nearest cluster center [11].

$$r_{nk} = \begin{cases} 1, \text{if } k = argmin_j ||x_n - \mu_j||^2 \\ 0, \text{otherwise} \end{cases}$$

Maximization step: Each cluster is recomputed to be the mean of the points assigned to the corresponding cluster. We minimize J with respect to k, keeping r_{nk} fixed [11]. Setting its derivative to zero yields

$$2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0.$$

Solving for μ_k we have

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}.$$

The convergence criteria are met when the algorithm no longer changes. However, finding the optimum is not guaranteed.

Mixtures of Gaussian (MoG) are motivated as a simple linear superposition of Gaussian components to provide a richer density model than the single Gaussian [11]. MoG was implemented using the Scikit-Learn library [12] in Python.

D. Supervised Learning Methods

We train a Support Vector Machine (SVM) and Random Forest (RF) for the supervised comparison element. The goal of SVM is to construct a hyperplane to use for classification. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class since the larger the margin, the lower the generalization error of the classifier [13]. Random Forests operate by constructing multiple decision trees at training time and outputting the class that is the mode of the classes for classification [14].

The SVM and RF models are implemented using the Scikit-Learn library [12] in Python, with no modifications made to the default arguments. We train/test split our data with an 80/20 split and a random state equal to 42. This configuration gives 35,911 training and 8,978 testing samples.

E. Reddit Bot

Classifying news headlines on the r/news and r/worldnews subreddits then commenting on the likelihood that a news headline is fake is accomplished by utilizing Reddit's API known as PRAW [15]. A class object with access to a Reddit accounts keys, username, and password are required to access writing comments to posts. Once a PRAW class is initialized, a stream of new submissions can be attained. The title of these submissions is the headline of the article due to the nature of the subreddits. The features of this title are extracted through python's Scikit-Learn function TfidfVectorizer [12] with 2000 max features. This vector is then classified using an MoG model with two components. The model is trained on the previously described dataset and then utilized to predict the probability and class of the headline. Each submission from the stream can then be replied to in the following format:

```
BEEP BOOP
We have identified this article to be:
(fake news/real news)
Fake news probability: ##
Real news probability: ##
Original headline title classification:
(Original title inputted to the program)
```

V. Experimental Results

For Scikit-Learn implementations, we classify each algorithm with multiple feature selections, including TF-IDF with and without PCA and sentiment features (see Table II). We found that the supervised classifiers, SVM and RF, performed better without PCA and achieved similar results with and without sentiment features reaching maximal accuracy scores of 95.17% and 94.17%, respectively. We determined K-means performance with label information to match clusters to label identity based on majority membership. We found the unsupervised classifiers, K-means and MoG, performed better without PCA with only TF-IDF features and performed better with PCA when sentiment features were included with TF-IDF but performed better overall with TF-IDF features at 86.6% and 84.36%, respectively.

TABLE II. MODEL PERFORMANCE

	Feature Selection			
	TF-IDF		TF-IDF + Sentiments	
Model	With PCA	Without PCA	With PCA	Without PCA
SVM	92.35%	**95.17%**	92.43%	95.12%
RF	93.33%	94.13%	93.34%	**94.17%**
K-Means	86.20%	**86.60%**	64.90%	64.70%
MoG	72.89%	**84.36%**	69.16%	65.68%

A. K-Means

To determine which combinations of features would produce the best K-means accuracy, we calculated accuracies for unique combinations for each feature across the 2D and 3D space. We found that the features that achieve the best accuracy (69.85%) are compound, positive, negative, neutral, and number of words. When applying K-Means to features derived from sentiment analysis, a lack of separability resulted in weak clusters and accuracies within the range of 60% - 70% (Figure I and Figure II). When applied to a six-dimensional space (i.e., features=Negative Sentiment Score, Positive Sentiment Score, Neutral Sentiment Score, Compound Sentiment Score, Number of Words, Number of Characters), under the Scikit-Learn implementation (random seed=0), K-means achieved a maximum accuracy score of 64.9%. When the dataset was shortened to only include the features Negative Sentiment Score, Compound Sentiment Score, Number of Words, the maximum accuracy score increased to 74.9% (random seed=12) (Figure III).

FIGURE I.

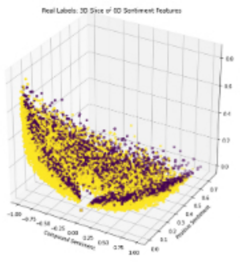


FIGURE II.

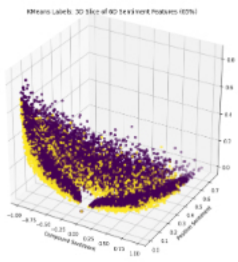
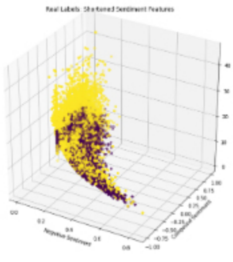


FIGURE III.



To explore our models against high-dimensional data to increase our accuracy, we calculated TF-IDF for our dataset, resulting in 2000 features for exploration. However, despite the inclusion of TF-IDF, a lack of separability in our data was still prevalent. This lack of separability produced weak clusters, with an average accuracy of around 60%-70%—similar to the results without TF-IDF. However, after trying several different random seeds to reproduce results achieved with our Bishop-model K-means, we found that Scikit-Learn implementation of K-Means could classify the 2000-dimensional dataset with 86.6% accuracy when the random seed was set to 40. To reduce the dimensionality, PCA was applied to our TF-IDF dataset to view more condensed plots (Figure IV and V). Despite the application of PCA, we can see that the separability did not improve, and the accuracy was affected by an insignificant amount. We applied PCA across a range of N values from 1 to 2000, iterating for each value of 100 features. Given this, we determined that the best accuracy would be equal to 86.3%, yielded by N = 300.

Figure IV.

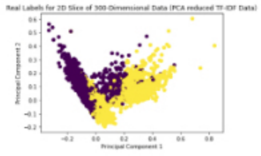
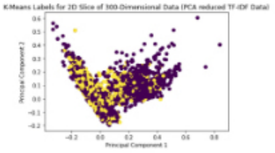


Figure V.



In more attempts to increase accuracy, we decided to apply PCA against both the sentiment dataset and the TF-IDF dataset (Figure VI, VII, VIII, and IX).

Figure VI.

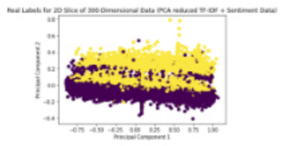


Figure VII.

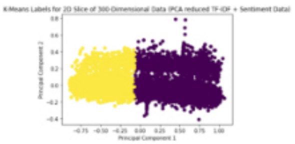


Figure VIII.

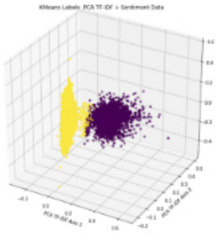
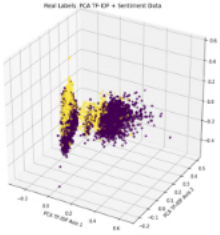


Figure IX.



VI. Discussion

A significant challenge for fake news classification with K-means is cluster separability. When applied to only sentiment analysis features, K-means resulted in weak cluster separability and weak fake news detection. When using K-means on all 2000 TF-IDF features, K-means achieved maximal accuracy; however, it is challenging to visualize clusters in a 2000 dimensional space.

Initially, the intention was to classify Facebook posts since it has a much larger user base and a more prominent Fake News issue, but Facebook doesn't permit access to its API. When selecting which submissions to classify on Reddit, the API gave access to recently popular, most popular of the past day/week/month/all time, and new submissions. Due to the sheer volume of comments, new comments on popular posts would not get noticed. The best approach is to classify an article before it rises to prominence to allow users who upvote it access to the data provided by the bot. In the future, the bot may be deployed to comments within more subreddits' comment sections which would require data scraping every link posted in a submission's comments, classifying the link as a news article, then extracting the headline of the article. User feedback by responding to the bots classifications with "good bot" or "bad bot" to indicate if the classification is correct or not could be utilized to improve the model as well. Adding misclassified data to the original training dataset would likely enhance the model but may be abused by users.

VII. Contributions

A. [redacted]

- Reddit Bot
- Paper (Reddit/Reddit Bot in sections I., IV., and VI.)

B. [redacted]

- K-Means Manual Implementation
- K-Means Scikit-Learn
- PCA Scikit-Learn
- Paper (K-means in section V.)
- Video Editing

C. [redacted]

- Data Pre-Processing
- Sentiment Analysis
- Paper (all sections)
- Latex
- Presentation Slides

D. [redacted]

- Paper (all sections)
- Background/Research
- Latex
- Presentation Slides

E. [redacted]

- K-Means Manual Implementation
- K-Means Scikit-Learn Implementation
- PCA Scikit-Learn
- Paper (K-means in section V.)

F. Michael Baluja

- Paper (Motivation in section I.)
- Paper (Supervised Methods in section IV.)
- PCA Manual Implementation
- Supervised Implementation

VIII. Code

The following link is to the GitHub repository containing all of the code utilized in this project.

Link to GitHub:

https://github.com/michaelbaluja/118bfinalproject

References

[1] C. Zhang, A. Gupta, C. Kauten, A. Deokar, and X. Qin, "Detecting fake news for reducing misinformation risks using analytics approaches," *European Journal of Operational Research*, vol. 279, 06 2019.

[2] G. Gravanis, A. Vakali, K. Diamantaras, and P. Karadais, "Behind the cues: A benchmarking study for fake news detection," *Expert Systems with Applications*, vol. 128, 03 2019.

[3] A. Bondielli and F. Marcelloni, "A survey on fake news and rumour detection techniques," *Information Sciences*, vol. 497, pp. 38–55, 2019. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1114201506

[4] Y. Chen, N. Conroy, and V. Rubin, "News in an online world: The need for an " automatic crap detector "," vol. 6?10, 10 2015.

[5] N. K. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015. [Online]. Available: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/pra2.2015.145052010082

[6] K. M. Yazdi, A. M. Yazdi, S. Khodayi, J. Hou, W. Zhou, and S. Saedy, "Improving fake news detection using k-means and support vector machine approaches," *International Journal of Electronics and Communication Engineering*, vol. 14, no. 2, pp. 38 – 42, 2020. [Online]. Available: https://publications.waset.org/vol/158

[7] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake news using text classification," *Security and Privacy*, vol. 1, p. e9, 12 2017.

[8] H. Ahmed, I. Traoré, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *ISDDC*, 2017.

[9] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[10] A. Rajaraman and J. D. Ullman, *Data Mining.* Cambridge University Press, 2011, p. 1–17.

[11] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics).* Berlin, Heidelberg: Springer-Verlag, 2006.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[14] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[15] B. Boe, *PRAW: The Python Reddit API Wrapper.* Bryce Boe Revision, 2021. [Online]. Available: https://github.com/praw-dev/praw/