How Much Bullshit Do We Need? Benchmarking Classical Machine Learning for Fake News Classification

Yudhanjaya Wijeratne
LIRNEasia, 12 Balcombe Place, Colombo, Sri Lanka (yudhanjaya@lirneasia.net)

Abstract

In a practical experiment, we benchmark five common text classification algorithms - Naive Bayes, Logistic Regression, Support Vector Machines, Random Forests, and eXtreme Gradient Boosting - on multiple misinformation datasets, accounting for both data-rich and data-poor environments. We test these methods by repeatedly reducing the sizes of training data, thus creating 435 AI models in total. From these models we make observations on the data requirements, the training times that such models might require in practice, and how the availability on these things impact accuracy and algorithm choices in practical scenarios. We then discuss the implications and avenues of further research.

Keywords: machine learning, misinformation, text classification, natural language processing

Introduction

The attention given to both misinformation (colloquially: "fake news") and countering it has increased dramatically since 2016. In computer science, this has lent itself to a number of ways of automating the detection of misinformation (Zhou & Zafarani, 2020[1]) - so-called "AI".

Of these techniques, we concern ourselves with the classification of pieces of text via linguistic features. This field broadly divides itself into what we might term classical machine learning and deep learning; one, a group of older methods that can function on smaller datasets; the other, a newer, far more sophisticated set of methods involving multi-layer neural networks and large datasets[2]. Both, as of current literature, demonstrate exceptional accuracy, often in the 90%-and-above range, far ahead of human control groups.

However, in practical use, accuracy is not the only consideration for task fitness of a machine learning algorithm. Two others are data, human effort and processing runtime, and often a Goldilocks-like decision criteria may be more useful than raw accuracy. This brings us to difficulties with current literature: one paper may use several hundred news articles, while another may use a hundred thousand tweets.

Thus there is a general lack of understanding of the amount of training data required for the task. While practitioners hold to a rule of thumb that that increasing the amount of training data generally increases the accuracy of most classification models, we do not know how much data is enough for a practical deployment of this technology; in the rare cases where this information exists, something that works for one particular test dataset may yield markedly different results on another, and so we are rarely able to reliably compare how different algorithms work at different sizes of different data, and where scaling stops, and the return on investment diminishes.

These factors are key to evaluating the fitness of a particular algorithm or method for use in the field, especially the latter: the operating costs of hardware and liveware guides decisions.

Thus, we ask a series of practical questions: of the commonly-cited classical machine learning algorithms, what levels of performance can we achieve, how much data do we need, and at what size of data does the performance of each algorithm begin to decay significantly? How much time do each of these algorithms require for their training? Where does scaling stop, and the return on investment diminish?

Methodology

To answer these questions, we demonstrate a simple experiment. We take five of the most commonly cited algorithms from the classical machine learning stack described above. We test them on four misinformation datasets, using a range of data and data sizes to mimic the spectrum between data richness and data sparsity, observing their accuracy and processing times, while controlling for the amount of human effort put into the experiment. We thus derive 435 models in total, using the same hardware, using the same programming languages processes.

We then observe, with these results the performance of different algorithms, the conditions of their task fitness, and thus attempt to answer the question about the difficulty of their use.

Data selection

As we have seen in the literature review, the size and shape of datasets change performance figures. Therefore, even when considering a straightforward use case, it is useful to make observations across a number of datasets. We therefore use four.

1) A class-balanced dataset comprised of 500,000 news articles extracted from the Fake News Corpus by Szpakowski (2020)[1] [3], which in turn is a collection of English news articles extracted using a typology from the (now-defunct) Open Sources project[2]. Our reduced version, hereafter referred to as FNC500k, is labelled *reliable* and *fake*: the reliable data is drawn directly from the *Credible* category presented in Szpakowski's source dataset, the fake category drawing equally from the *Fake News* and *Conspiracy Theory* categories. Both this source and the format of a binary classification is in use in the Kaggle Fake News Detection Challenge KDD 2020[3].

2) The dataset from the Kaggle Fake News Detection Challenge KDD 2020[4], which includes 20,800 news articles curated by K. Shu [4]. This dataset (henceforth referred to as the Kaggle dataset) is part of fake news detection shared task for the Second International TrueFact Workshop: Making a Credible Web for Tomorrow. The data is labelled as fake (*1*) or real (*0*) respectively.

3) The dataset by George McIntire, commonly cited as the KDNuggets Fake News dataset[5] [5], consisting of news articles labelled as *real* and *fake*. The fake component here draws from a previously released Kaggle dataset assembled from sources flagged by BS Detector; the real is derived from AllSides.com. This dataset is mentioned as having 10,558 articles, but public forks of it[6], duplicated from McIntire's Github repository, contain only 6335 entries.

[1] https://github.com/several27/FakeNewsCorpus
[2] https://github.com/BigMcLargeHuge/opensources
[3] https://www.kaggle.com/c/ds2020/data
[4] https://www.kaggle.com/c/fakenewskdd2020/data
[5] https://www.kdnuggets.com/2017/04/machine-learning-fake-news-accuracy.html
[6] https://www.kaggle.com/mrisdal/fake-news

4) The LIAR dataset by Wang (2017) [6]: 12,800 short statements from Politifact.com, among them excerpts from news releases, TV/radio interviews, campaign speeches, TV advertisements, social media posts, and statements issued in political debate; these are labelled as *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*.

Three of these datasets reduce to of truth to a binary categorization. This approach is commonly used categories for these types of work, such as in the KDNuggets and Kaggle data, or in (Gravanis et al., 2019; Monteiro et al., 2018), or variations of the theme such as fake vs truthful used by (Ahmed & Saad, 2017), misinformation vs truth information (Yu et al., 2017) and unreliable vs reliable (Kaliyar et al., 2020). We use the LIAR dataset to represent more nuanced multi-class ontologies.

Data pre-processing

The datasets were cleaned to eliminate: HTML, XML and other web tags; punctuation; multiple whitespaces; numbers; stopwords; words containing 3 or less characters (commonly acronyms). All text was transformed to lowercase.

Model training

First, the datasets are loaded, randomly resampled, then partitioned according to an 80:20 training:testing split. The text content of these datasets were then converted into a matrix of unigrams ranked by their inverse frequency within a document - commonly known as a TF-IDF (Term Frequency-Inverse Document Frequency) matrix [7]; this follows the logic demonstrated by Ahmed et al (2017) [8] in their assertion that unigrams and term frequencies performed best as features for the algorithms they tested.

The algorithms we have selected are from the classical machine learning stack: Logistic Regression (LR), Naive Bayes (NB) [9], Random Forests (RF) [10], XGBoost (XGB) [11] and Support Vector Machines (SVM) [12]. The SVM in question is set to use the Radial Bias Function (rbf) kernel. We have selected these because, as shown in the literature, they are reliable and relatively simple, compared to deep learning approaches.

Now we must ascertain how much data these algorithms might require. In one version of Zeno's Paradox, Achilles chases a snail by repeatedly halving the distance between them. Likewise, we repeatedly halve the amount available for model training (and rounding down in instances where the halving process would give us half a document). For a lower bound, we stop this process once we have one dataset in the 300s. This entire process was repeated thrice, with a randomly sampled version of the source dataset being used for each run.

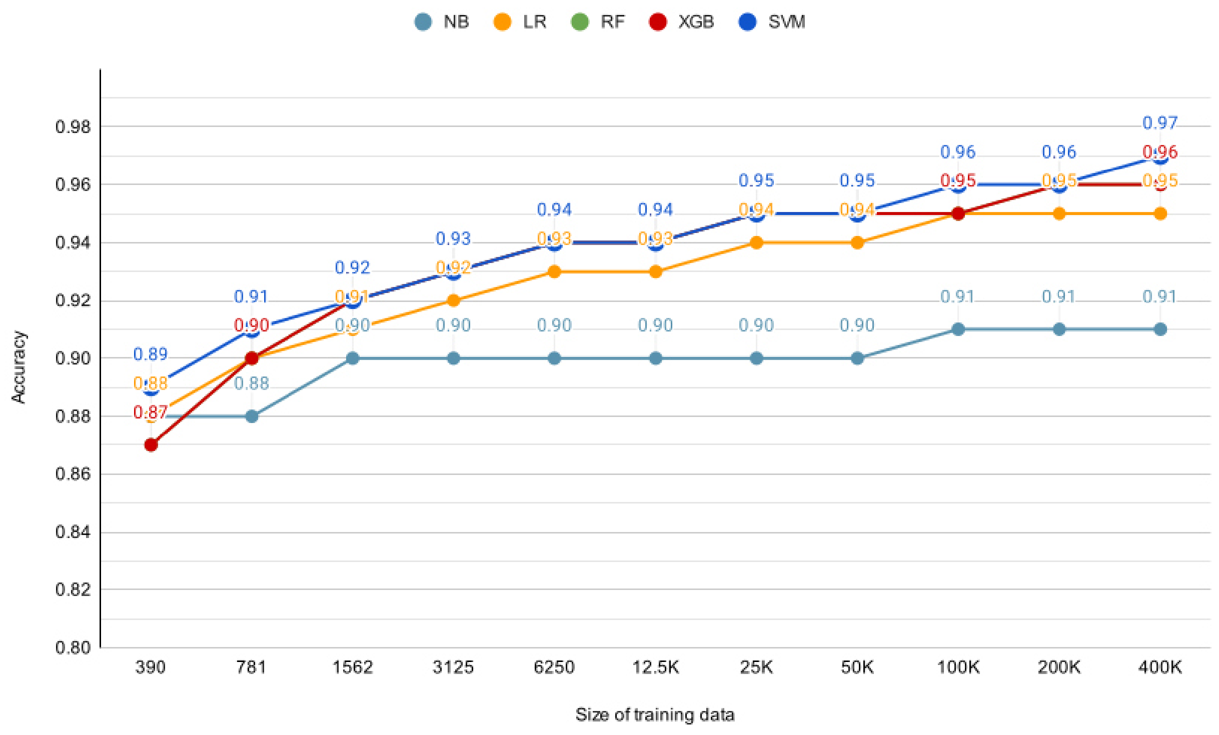
Thus, for FNC500K, we therefore generated 165 models in total. For the Kaggle dataset, 105; for the KDNuggets dataset, 75; for LIAR, we generated 90 models. This comes to a total of 435 models, which gives us enough averaged data that we can make reasonable observations with them.

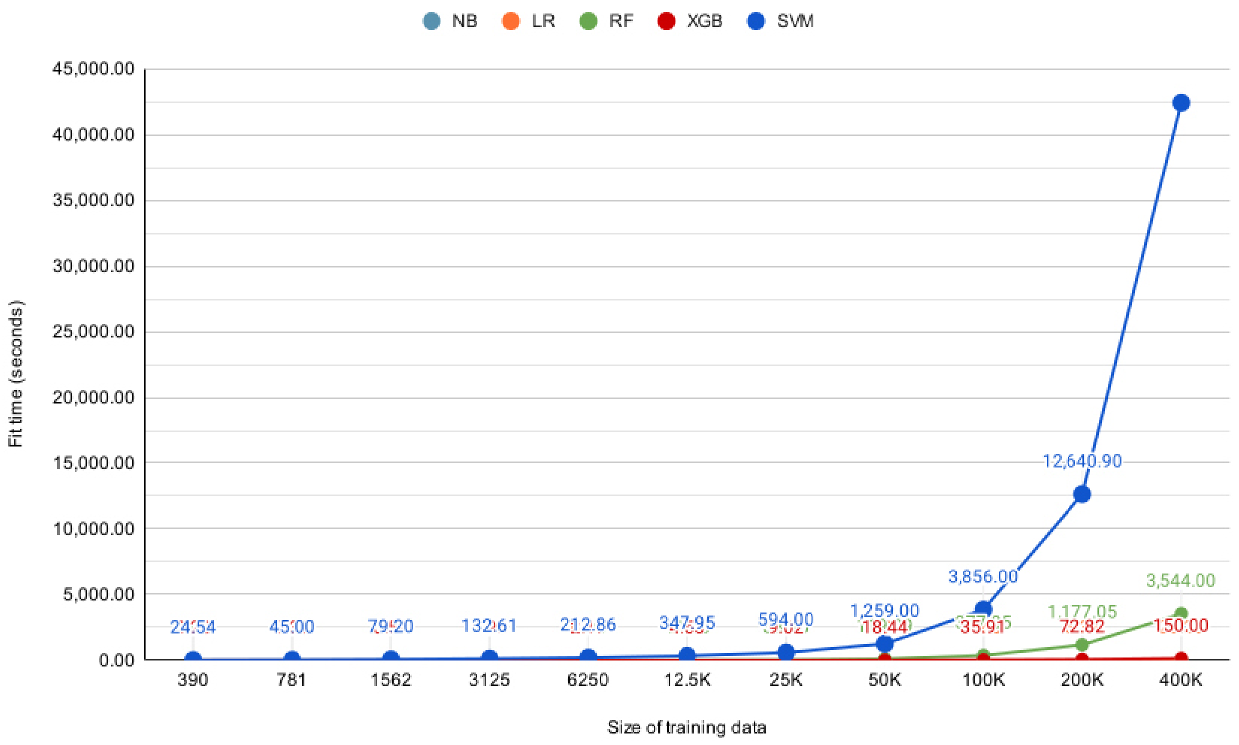
Each model that was generated was tasked with classifying the test dataset derived from the source at the start of the process. Both the accuracy (as the F1 score of the model) and the time taken to train each model was recorded and the results averaged over three training runs.

The training time was taken as the time required to complete the scikit-learn fit() method for each model (referred to in the charts as fit time). This is by no means a perfect measure of computational resources or the time taken to train a model, as the minutiae of interactions between various software implementations, libraries and optimizations can make it difficult to pinpoint these values with precision; however, these averaged values are of use when relative to each other.

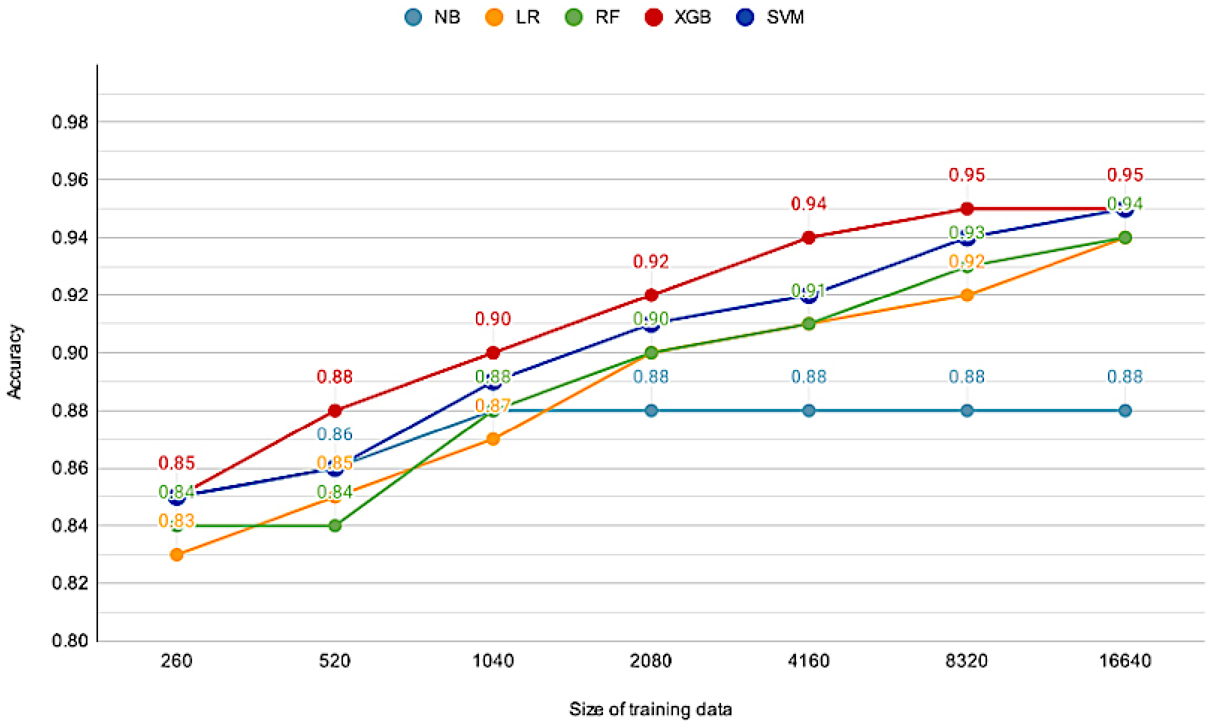
All these processes were performed using the same software stack (Windows 10 with updates and Windows Defender disabled; Python 3.8.3; gensim 3.8.3; scikit-learn 0.23.1; xgboost 1.2.0) and the same hardware (AMD Ryzen 3600 - 6 cores, 12 threads; 32 GB DRR 4 RAM at 3.2 Ghz). All data was loaded into RAM before processing to minimize data fetch delays that could be introduced by hard drives or SSDs. One human was used for all these tasks.

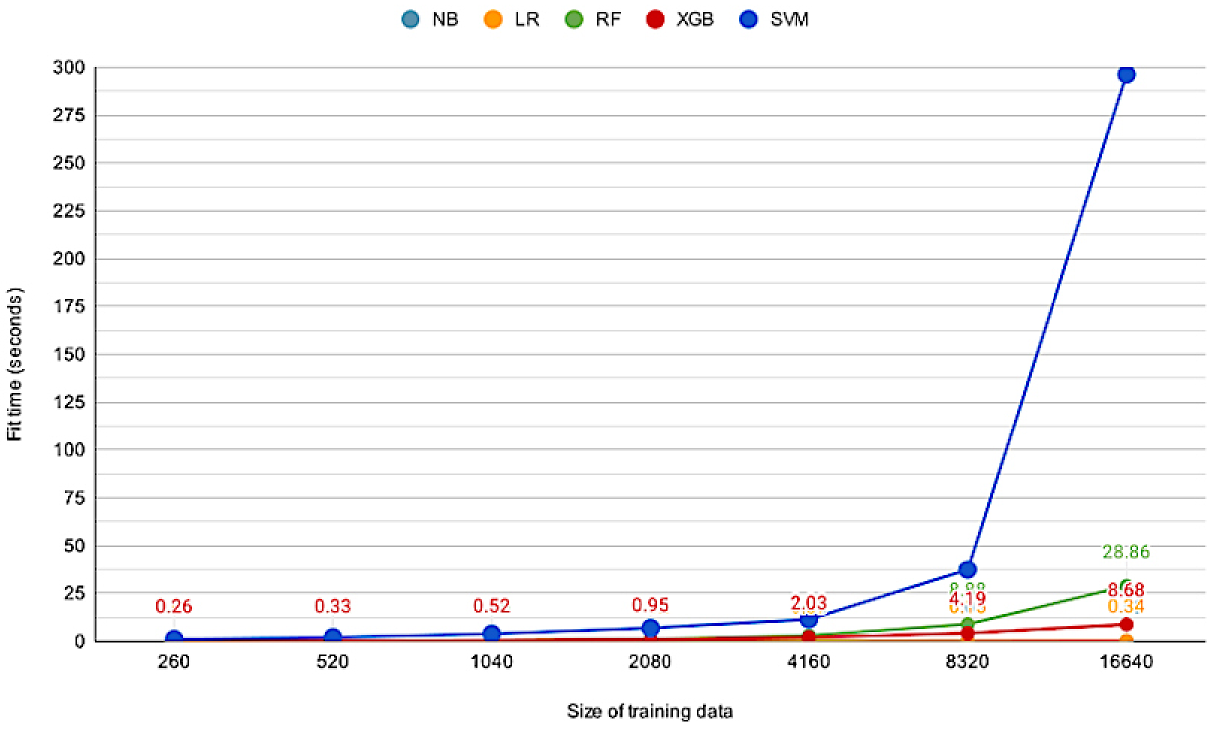
Results, FNC500k



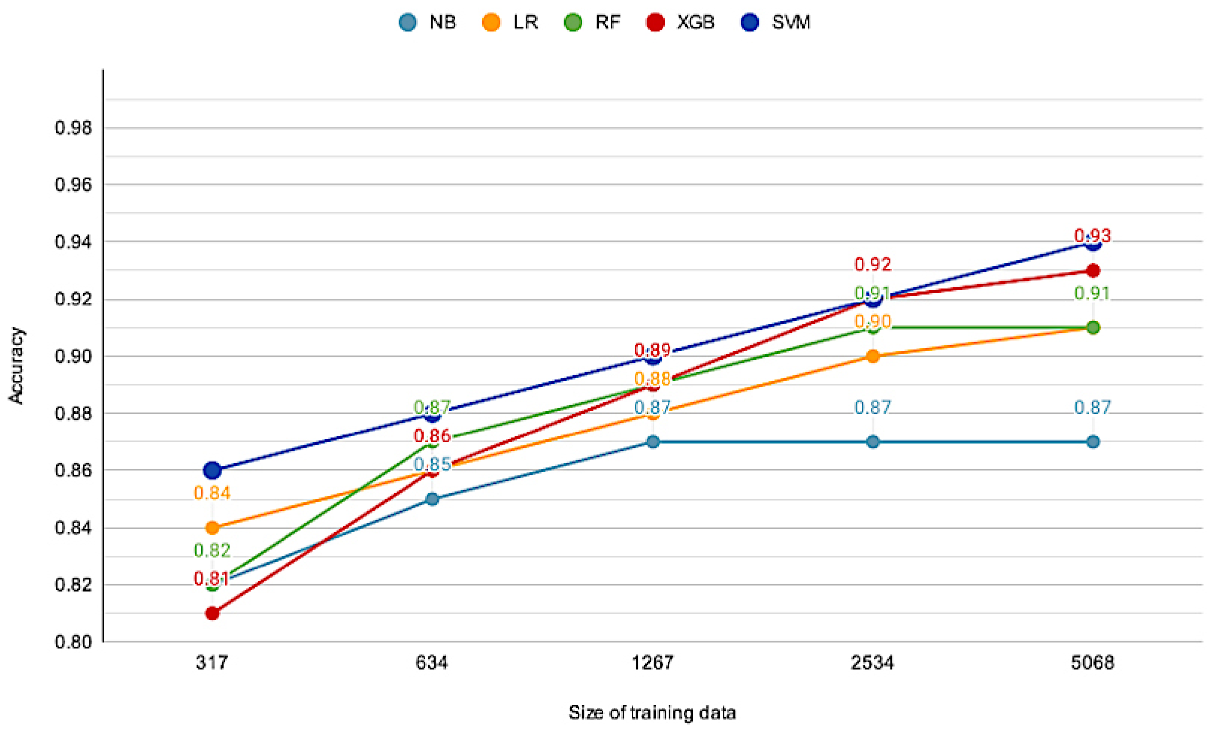


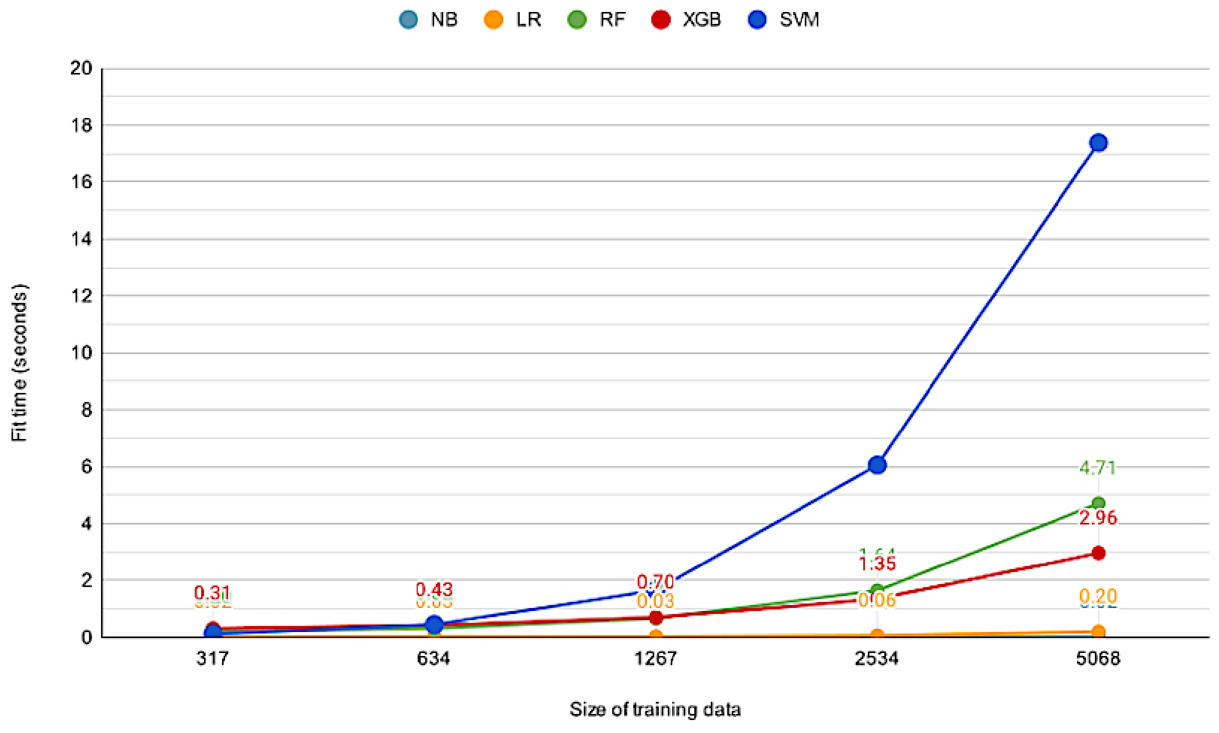
Results, Kaggle



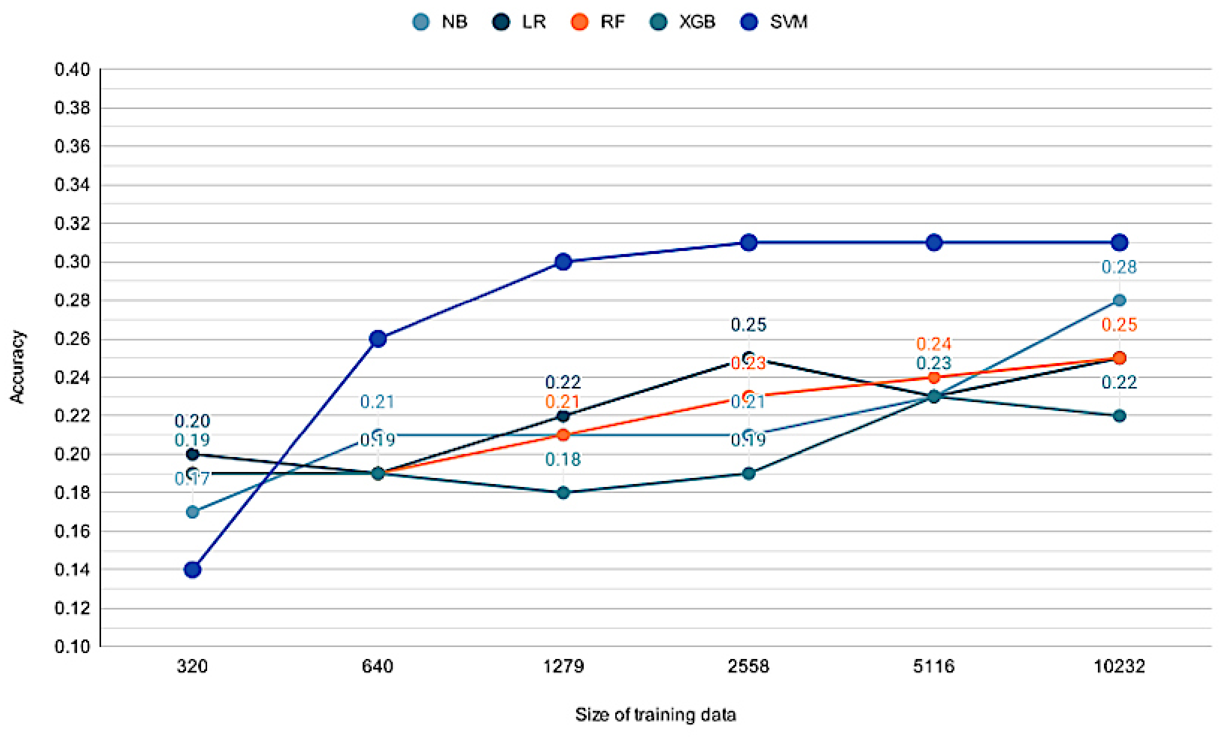


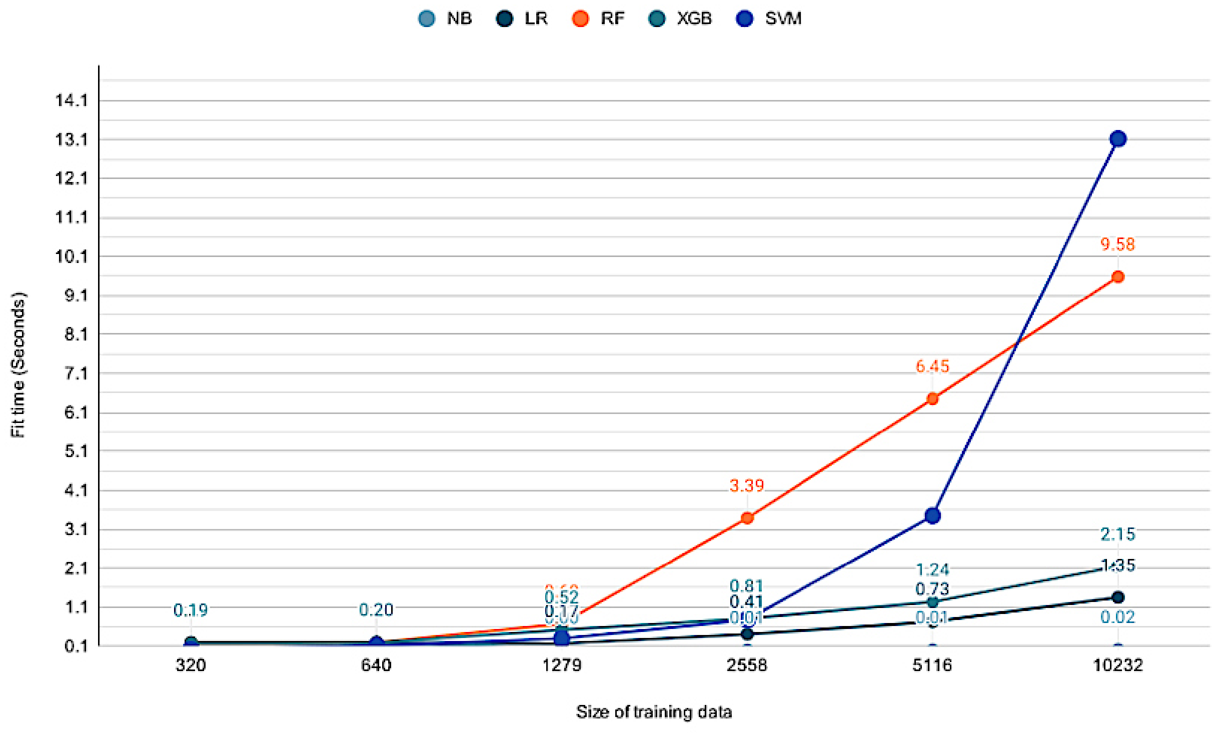
Results, KDNuggets





Results, LIAR





Discussion of results

We can here observe a few things that hold true across the board. The general rule of thumb in machine learning is that more data yields better results; however, for those algorithms that do scale with more data, we can observe that we need exponential increases in dataset size for linear, incremental gains in accuracy. For our assumed role of a factchecker examining implementation, this relationship is an important one to understand. On the FNC500K dataset, for instance, a mere 1% gain in accuracy at the top end of data for most algorithms tested requires an additional 200,000 annotated samples.

SVMs repeatedly set themselves apart as the most performant at most sizes of training data; they are outperformed only on the Kaggle dataset. This accuracy tallies with observations from other literature (Ahmed et al, 2017 [8], Perez-Rosas et al, 2017 [13], Patwa et al, 2020 [14]) and a long-standing superiority attached to this method from as far back as Joachims (1998) [15].

However, despite their vaunted accuracy, the runtime data reveals that at no point can SVMs be called efficient when their fit times are called into question. At the lowest end on FNC500K, the SVM approach still clocked an average of 24.54 seconds - as compared to under 4 seconds for every other approach. At 400,000 samples, the SVM averaged a runtime of 42412.636 seconds as compared to the 150 seconds for the XGB approach, which was closest in terms of accuracy. This SVM implementation is based on Chang and Lin (2011)'s libSVM [16]; the 'fit time' scales at least quadratically[7]. While it is possible to optimize further, or to find implementations more suited to larger datasets, this issue of training time stands out as a general problem for SVMs.

Therefore, in data-rich environments where training time is key, an SVM approach may be one of the worst approaches to take. On the other hand, a data-poor environment may find SVMs to be of great value, as the fit time reduces exponentially.

The Naive Bayes algorithm consistently shows itself to be the fastest here. However, it generally records the lowest accuracy scores and exhibits poor scaling, exhibiting distinct accuracy plateaus in all four datasets.

Tree-based ensemble methods like Random Forests and XGB are, from these results, excellent general out-of-the-box solutions. XGB approaches typically match or slightly exceed RF for accuracy, and typically shadow SVMs within 1% of their accuracy figures, and both feats are achieved with with significantly lower training time.

Logistic Regression puts up a surprisingly efficient showing: consistently closest to the NB in time, but closer to RF and XGB accuracies given enough data. Panckevičius & Marcinkevičius (2017) [17] speak highly of Logistic Regression, although our results are not as overwhelmingly in favor.

Thus, despite the literature championing SVMs, a practical engineer, keeping in mind the scaling above, seems to have more options on their hands within the traditional machine learning stack. The general combination of performance and speed make XGB and similar approaches [18] useful as default algorithms. SVMs can be used if training times are irrelevant; Logistic Regression provides an attractive alternative in case training times and compute resources need to be prioritized.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

The traditional machine learning stack is even surprisingly performant on the multi-class data structure of the LIAR dataset. Girgis et al (2018) [19], approaching the same classification task and dataset with deep learning techniques, recorded accuracy results of 0.217 with Gated Recurrent Units (GRU) and 0.2166 with Long Short-Term Memory (LSTM) approaches - outdone by all models here. This tallies with results from Wang (2017) [6] and the LIAR paper's comparisons of SVMs and Logistic Regression versus Convolutional Neural Networks (CNN).

Overall, the takeaway is that given enough data, it is remarkably easy to create a classification model that is between above an arbitrary 90% mark for accuracy - far above the human baseline. In particular, SVMs, Logistic Regression and XGBoost show themselves to be capable over 90%-and-above accuracy in binary classification of misinformation, working with as low as 781 samples of training data (FNC500K), 2080 samples (Kaggle) and 2534 samples (KDNuggets).

And of course, this is a simple example: the deep learning stack is untouched here. Several candidates exist that we might wish to resurrect, albeit with greater effort. As-is, almost the entire program used to this experiment can be constructed out of ready-made functions freely and easily available in programming languages like Python. Getting to the highest possible level of accuracy will, of course, be a feat for academia to pursue for the conferences, but acceptable results are quite easy to achieve.

Acknowledgements

This research has been made possible through a grant from the Asia Foundation.

Bibliography

[1] X. Zhou and R. Zafarani, "A survey of fake news: Fundamental theories, detection methods, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–40, 2020.

[2] Y. Wijeratne and D. C. Attanayake, "Artificial intelligence for factchecking: Observations on the state and practicality of the art," 2021.

[3] M. Szpakowski, "Fake news corpus," 2020. [Online]. Available: https://github.com/several27/FakeNewsCorpus

[4] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD explorations newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[5] G. McIntire, "Fake news corpus," 2017. [Online]. Available: https://www.kdnuggets.com/2017/04/machine-learning-fake-news-accuracy.html

[6] W. Y. Wang, "" liar, liar pants on fire": A new benchmark dataset for fake news detection," *arXiv preprint arXiv:1705.00648*, 2017.

[7] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.

[8] H. Ahmed, I. Traore, and S. Saad, "Detection of online fake news using n-gram analysis and machine learning techniques," in *International conference on intelligent, secure, and dependable systems in distributed and cloud environments*. Springer, 2017, pp. 127–138.

[9] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[12] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[13] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," *arXiv preprint arXiv:1708.07104*, 2017.

[14] P. Patwa, S. Sharma, S. PYKL, V. Guptha, G. Kumari, M. S. Akhtar, A. Ekbal, A. Das, and T. Chakraborty, "Fighting an infodemic: Covid-19 fake news dataset," *arXiv preprint arXiv:2011.03327*, 2020.

[15] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[16] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[17] T. Pranckevičius and V. Marcinkevičius, "Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification," *Baltic Journal of Modern Computing*, vol. 5, no. 2, p. 221, 2017.

[18] E. Al Daoud, "Comparison between xgboost, lightgbm and catboost using a home credit dataset," *International Journal of Computer and Information Engineering*, vol. 13, no. 1, pp. 6–10, 2019.

[19] S. Girgis, E. Amer, and M. Gadallah, “Deep learning algorithms for detecting fake news in online text,” in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*. IEEE, 2018, pp. 93–97.