Supervised Learning Comparison For Binary Text Classification

Michael Baluja

March 25, 2021

Abstract

There are many popular classification algorithms in use, all with varying strengths. It can be difficult at times to understand which algorithm will lead to optimal performance for a given task and data set. We present an analysis of different algorithms used for text classification, namely Support Vector Machines, Logistic Regression, Random Forests, and Artificial Neural Networks. We also compare the algorithms presented across multiple performance metrics in order to understand the trade-off of using off-the-shelf models versus building a neural network.

1 Introduction

While there are many promising state-of-the-art neural networks capable of performing tasks such as text classification with nearly 95% accuracy on standard data sets, implementing and training these networks may often be computationally prohibitive [**5**]. This paper looks to understand the comparative effectiveness of different off-the-shelf[1] machine learning classification models against more complex neural networks, all trained on various text classification tasks. Through this research, we look to see the relative performance to help better understand whether or not it is necessary to implement and train neural networks for certain use cases.

There are myriad different classification tasks that fall under the category of text classification, and many different reasons to analyze the information present in these data. In this paper, we analyze model performance in terms of accuracy for three distinct tasks: binary sentiment analysis, binary clickbait classification and binary subjectivity/objectivity classification. All of the listed tasks are binary text classification tasks with a roughly even class distribution across small (10,000 - 32,000 sample) data sets.

[1]For this paper, we consider our off-the-shelf algorithms as the Support Vector Machine, Logistic Regression, and Random Forest, all implemented through the scikit-learn package.

2 Methodology

This report covers performance across three different data sets, four different classification algorithms, and four different error metrics, as listed. All parameter spaces are similar to what is done by Caruana et al., with the exception being the Random Forest [**4**].

2.1 Classification Algorithms

Support Vector Machines (SVM)

We train across linear, polynomial, and radial basis kernels. Our polynomials include degree two and degree three. Our radial basis kernel uses gamma values of 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2. Our C regularization ranges in multiples of 10 from 1e-7 to 1e3.

Logistic Regression (LogReg)

We train both unregularized and L2 regularized models. Our C regularization ranges in multiples of 10 from 1e-8 to 1e4.

Random Forest (RF)

We train a forest of 256, 512, 1024, 2048, 4096, and 8192 trees. The size of the feature set considered at each split is 1.[2]

2.1.1 Neural Networks (NN)

We train a Character-level Convolutional Neural Network with default hyper-parameter settings for computational purposes. This model utilizes stochastic gradient descent with a learning rate of 0.01, gamma and momentum of 0.9, and a batch size of 1024 samples over 100 training epochs. Further implementation information can be found at [**2**], [**6**].

[2]We stray from the parameter space utilized in [**4**] due to implementation. As the CountVectorizer utilized for our data sets only returns a sparse vector as a sigle feature, we choose to vary the number of trees for search instead of implementing additional features based on heuristics.

2.2 Error Metrics

As this report looks to understand the best algorithm to implement for given text classification tasks, we adopt error metrics that are fairly standard in the natural language processing literature. In this regard, we focus the performance attention on accuracy (ACC) and the F1 metric. However, we also recognize and understand that not all classification tasks will equally want to balance precision (PREC) and recall (REC) as is done with the F1 metric. For this reason, we include precision and recall metrics for each data set and classifier combination to allow the reader further analysis.

2.3 Data Sets

The three data sets considered for this task all fall into the category of binary text classification, although with different tasks, as described below.

Yelp Polarity (Yelp)

Sentiment analysis data set of yelp review polarity for positive or negative sentiment. While the original data set is comprised of 299,000 training/testing samples, both splits were combined and trimmed to 32,000 samples for computational accessibility. Class ratios were preserved with the split.

Clickbait (Click)

Data set containing news headlines that are considered to be or not be "clickbait". This set contains 16,000 samples each of clickbait and non clickbait headlines for a total of 32,000 total samples[**1**].

Subjectvity Objectivity (SubOb)

Data set containing sentences that are tagged to either contain subjective or objective information. The set contains 5,000 samples each of subjective and objective samples for a total of 10,000 total samples.

3 Experiment

We utilize a 5-fold cross validation for hyper-parameter tuning to select the optimal model for each off-the-shelf algorithm we include. Following the work of Caruana et al., we perform this validation over 5000 training and validation samples, with each cross validation trial including 4000 training and 1000 validation samples [**4**]. The rest of the data is set aside for testing the performance of our final optimal model. The model with the best performance on the validation data is presented in Tables 1 - 6. Implementation details are present in Appendix B.

In terms of data transformation, we utilize a character-level vectorized data form for our SVM, LogReg, and RF models to complement the character level implementation of our NN. While some prefer to utilize a maximum feature ceiling of 10% of the feature space, initial testing found performance increases when not imposing such a limit.

For our neural network, we adopt a Character-level Convolutional Neural Network from Zhang et al, with unofficial code provided by Mehrani in [**2**]. We utilize the default hyper-parameter settings and do not perform any sort of tuning for computational reasons.

Due to the variance from the random sampling that occurs when selecting training data, we perform each trial of data splitting, hyper-parameter tuning (with the exception of the NN), and model evaluation three times. We aggregate the data from these trials to report on the listed error metrics, and also use the variance between these trials to understand the significance of performance difference between our different data set and algorithm combinations.

Mean trial performance for each combination of model and data set is shown in Table 1 for test performance and Table 3 for train performance. Mean trial performance for each model over all data sets is shown in Table 2 for test performance. Note that the best performance per metric is in **bold**, and performance differences that do not have a statistically significant difference from the best performance (through a two sample t-test) at $p < 0.05$ are marked with a *, with p-values present in A.2.1 and A.2.2.

Evaluating the mean test performance in Table 1, we see the best performance and all significant results come from our neural network. For all other cases, performance is in the neighbourhood of 50% to 60%, signifying that the models did poorly on discriminating on the binary classification tasks, possibly due to extremely non-seperable data. However, we do see continually higher performance on the clickbait data set than on the yelp and subjectivity objectivity data sets. This poor performance is still present when restricting the feature space as mentioned earlier.

Comparing across metrics in Tables 1 and 2 we see that in most cases, performance across accuracy, precision, recall, and F1 are all within a few percent of each other. It is important to note, however, that low performance for the precision, recall, and F1 metrics in some instances is caused by a lack of the classifier predicting any values for either the positive or negative class, resulting in a score of 0 for those metrics for a given training cycle. This occurred in the neural network for the Yelp data set, as seen in A.1.

Table 1: Mean Test Performance by Model and Data Set

Model	Data Set	ACC	PREC	REC	F1
SVM	Yelp	.558	.559	.557	.554
SVM	Click	.654	.659	.654	.651
SVM	SubOb	.541	.573	.541	.485
LogReg	Yelp	.556	.561	.555	.545
LogReg	Click	.654	.658	.654	.653
LogReg	SubOb	.535	.549	.537	.504
RF	Yelp	.525	.525	.525	.523
RF	Click	.655	.659	.656	.652
RF	SubOb	.527	.537	.529	.500
NN	Yelp	.514	.347	.542*	.415*
NN	Click	**.895**	**.905**	.885*	**.895**
NN	SubOb	.591	.598	**.916**	.702

Table 2: Mean Test Performance by Model

Model	ACC	PREC	REC	F1
SVM	.584*	.597*	.584*	.564*
LogReg	.582*	.589*	.582*	.567*
RF	.569*	.574*	.569*	.559*
NN	**.667**	**.616**	**.781**	**.670**

This trend of neural networks achieving highest performance is similar in Table 2, with all top performance being achieved by neural networks. However, when averaging over all results we see significant performance by other algorithms. This implies that overall, neural networks do not deliver significant performance increases when compared to any of our tested off-the-shelf algorithms.

Table 3 shows mean test performance for each model and data set combination. When comparing these results with the results seen in Table 1, we only see large differences in performance when looking at the neural network. This indicates that while the off-the-shelf algorithms are unable to fully solve the classification problem, the slightly above-chance performance they do have generalizes well.

4 Discussion

4.1 Hyper-parameter Analysis

We see heat maps for validation performance across different hyper-parameters for each off-the-shelf model in Appendix A.3. From this, we see an overall trend of high performance arising from situations in which the C value is high. In Figure A.3.2 we see an overall trend of roughly similar performance (note the accuracy scale), but best performance from Caruana et al.'s original tree count of 1024 [**4**]. Figure A.3.4 shows that we get higher performance from higher polynomial degrees, which gives insight to possibly training models on higher degree polynomial models. We also see in A.3.5 that best performance tends to come from a small radial width with a mid-range to high C value. We also note the relatively small range of accuracy that these different parameters achieve.

4.2 Conclusion

To conclude, there are a few points worth acknowledging. Answering the question of whether or not it is worthwhile to train a neural network instead of an off-the-shelf algorithm depends on time and computational speed constraints. While we do see neural networks outperforming all of the off-the-shelf algorithms, we note that the neural networks took an average of 15-20 minutes for training a single model, while grid searches for the LogReg and RF models would be completed in seconds to minutes. It is also important to note that while our off-the-shelf algorithms were trained using CPUs, the neural networks were trained using a CUDA-enabled GPU. In the case of difficult problems as present in the yelp and subjectivity/objectivity data sets, it may likely be necessary to utilize neural networks when possible in order to boost performance. While the performance differences may not be statistically significant, they may still prove to have real-world significance.

It is important to mention the overall poor performance for all models, with the exception of the neural network trained on the clickbait data set. Additional testing was conducted using the same test bench with the LETTER data set of the original paper to ensure test bench functionality. Performance

Table 3: Mean Train Performance by Model and Data Set

Model	Data Set	ACC	PREC	REC	F1
SVM	Yelp	.576	.578	.576	.573
SVM	Click	.654	.659	.655	.653
SVM	SubOb	.542	.574	.541	.485
LogReg	Yelp	.564	.567	.562	.552
LogReg	Click	.654	.657	.653	.652
LogReg	SubOb	.539	.552	.537	.507
RF	Yelp	.627	.628	.626	.625
RF	Click	.652	.656	.651	.649
RF	SubOb	.555	.570	.554	.528
NN	Yelp	.517	.516	.576	.544
NN	Click	**1**	**1**	**.999**	**1**
NN	SubOb	.665	.664	.610	.635

on this trial reached up to 96.34% accuracy on validation data, indicating that our poor performance is due to model choice for the individual data sets as opposed to implementation. One possible explanation for poor performance is the low number of samples used for training. When using the full yelp data set without limitation, many models, such as the neural network used in this work are able to reach upward of 95% accuracy. Additional training cycles were completing with more samples in the yelp data set, but no significant improvement in performance was noted.

Another important point pertains to the high similarity of the different error metrics when looking at the performance of the off-the-shelf algorithms. However, there is more variability in the different metrics when looking at our neural network results in Tables 1 and 2. One possible explanation for this may have to due with the balance of our data sets, but further work to investigate performance across different metrics for unbalanced data sets should be done.

We believe it is necessary for future research to utilize ensemble methods for comparison against neural networks. In addition, heuristic-based feature addition may prove beneficial for models such as the random forest, and should be considered in future work.

References

[1] Chakraborty, A., Paranjape, B., Kakarla, S. and Ganguly, N., 2016. Stop Click-bait: Detecting and Preventing Clickbaits in Online News Media. Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining,.

[2] ArdalanM, "ArdalanM/nlp-benchmarks," GitHub. [Online]. Available: https://github.com/ArdalanM/nlp-benchmarks/blob/master/src/cnn/ [Accessed: 04-May-2020].

[3] M. Baluja, "michaelbaluja/classifiercomparison," GitHub, Dec-2020. [Online]. Available: https://github.com/michaelbaluja/classifiercomparison. [Accessed: 14-Dec-2020].

[4] R. Caruana and A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," International Conference on Machine Learning, vol. 23, 2006.

[5] S. Ruder, "NLP Progress: English Text Classification," NLP. [Online]. Available: http://nlpprogress.com/english/text_classification.html. [Accessed: 03-Dec-2020].

[6] X. Zhang, J. Zhao, and Y. LeCunn, "Character-level Convolutional Networks for Text Classification," 2015.

A Additional Results

A.1 Raw Test Set Scores

We present raw test data performance in A.1.

A.1: Raw Test Performance by Model and Data Set

Model	Data Set	Metric	Values
SVM	Yelp	ACC	[0.567, 0.544, 0.564]
SVM	Yelp	PREC	[0.567, 0.545, 0.566]
SVM	Yelp	REC	[0.567, 0.543, 0.562]
SVM	Yelp	F1	[0.567, 0.539, 0.556]
LogReg	Yelp	ACC	[0.556, 0.560, 0.551]
LogReg	Yelp	PREC	[0.562, 0.562, 0.559]
LogReg	Yelp	REC	[0.556, 0.559, 0.551]
LogReg	Yelp	F1	[0.545, 0.555, 0.535]
RF	Yelp	ACC	[0.527, 0.527, 0.520]
RF	Yelp	PREC	[0.527, 0.528, 0.520]
RF	Yelp	REC	[0.527, 0.527, 0.520]
RF	Yelp	F1	[0.526, 0.524, 0.520]
NN	Yelp	ACC	[0.555, 0.493, 0.493]
NN	Yelp	PREC	[0.548, 0.000, 1.000]
NN	Yelp	REC	[0.625, 0.000, 0.493]
NN	Yelp	F1	[0.584, 0.000, 0.660]
SVM	SubOb	ACC	[0.543, 0.537, 0.541]
SVM	SubOb	PREC	[0.582, 0.570, 0.567]
SVM	SubOb	REC	[0.544, 0.540, 0.539]
SVM	SubOb	F1	[0.485, 0.484, 0.488]
LogReg	SubOb	ACC	[0.533, 0.534, 0.537]
LogReg	SubOb	PREC	[0.556, 0.548, 0.545]
LogReg	SubOb	REC	[0.536, 0.535, 0.538]
LogReg	SubOb	F1	[0.489, 0.503, 0.520]
RF	SubOb	ACC	[0.518, 0.534, 0.528]
RF	SubOb	PREC	[0.534, 0.541, 0.536]
RF	SubOb	REC	[0.523, 0.535, 0.529]
RF	SubOb	F1	[0.479, 0.517, 0.505]
NN	SubOb	ACC	[0.551, 0.772, 0.501]
NN	SubOb	PREC	[0.501, 0.791, 0.501]
NN	SubOb	REC	[1.000, 0.748, 1.000]
NN	SubOb	F1	[0.668, 0.769, 0.668]
SVM	Clickbait	ACC	[0.655, 0.654, 0.653]
SVM	Clickbait	PREC	[0.657, 0.660, 0.659]
SVM	Clickbait	REC	[0.655, 0.654, 0.652]
SVM	Clickbait	F1	[0.654, 0.650, 0.649]
LogReg	Clickbait	ACC	[0.654, 0.655, 0.654]
LogReg	Clickbait	PREC	[0.656, 0.656, 0.660]
LogReg	Clickbait	REC	[0.654, 0.655, 0.654]
LogReg	Clickbait	F1	[0.653, 0.654, 0.651]
RF	Clickbait	ACC	[0.656, 0.656, 0.653]
RF	Clickbait	PREC	[0.662, 0.657, 0.659]
RF	Clickbait	REC	[0.656, 0.656, 0.653]
RF	Clickbait	F1	[0.653, 0.655, 0.649]
NN	Clickbait	ACC	[0.875, 0.904, 0.907]
NN	Clickbait	PREC	[0.880, 0.909, 0.925]
NN	Clickbait	REC	[0.870, 0.899, 0.887]
NN	Clickbait	F1	[0.875, 0.904, 0.906]

A.2 Table 1 Comparison p-values

We present p-values for the uncorrected two-sample t-tests done in Table 1 and 2. These results are present in A.2.1 and A.2.2, respectively.

A.2.1: p-values for Table 1 Comparisons

Model	Data Set	ACC	PREC	REC	F1
SVM	Yelp	2.171e-6	9.124e-7	0.000	1.834e-5
SVM	Click	1.914e-5	4.874e-5	0.036	1.776e-5
SVM	SubOb	4.353e-6	1.854e-5	0.011	2.194e-6
LogReg	Yelp	2.839e-6	1.559e-6	0.000	1.033e-5
LogReg	Click	1.906e-5	4.871e-5	0.036	1.761e-5
LogReg	SubOb	3.947e-6	1.292e-5	0.011	8.281e-6
RF	Yelp	7.654e-6	6.999e-6	0.000	1.444e-5
RF	Click	1.957e-5	4.969e-5	0.036	1.806e-5
RF	SubOb	5.127e-6	1.037e-5	0.010	1.240e-5
NN	Yelp	0.075	0.135	0.504	0.206
NN	Click	1.000	1.000	0.735	1.000
NN	SubOb	0.029	0.035	1.000	0.005

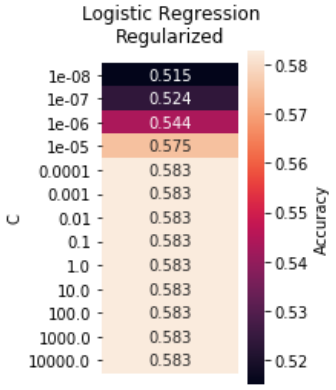
A.2.2: p-values for Table 2 Comparisons

Model	ACC	PREC	REC	F1
SVM	0.234	0.848	0.086	0.281
LogReg	0.220	0.790	0.084	0.295
RF	0.167	0.679	0.069	0.259
NN	1	1	1	1

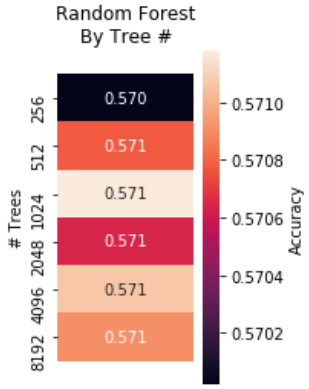
A.3 Hyper-parameter Heat Maps

The following graphs show validation set performance during hyper-parameter selection.

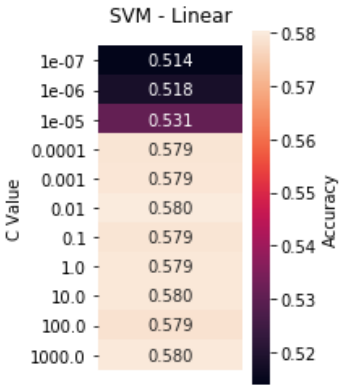
A.3.1



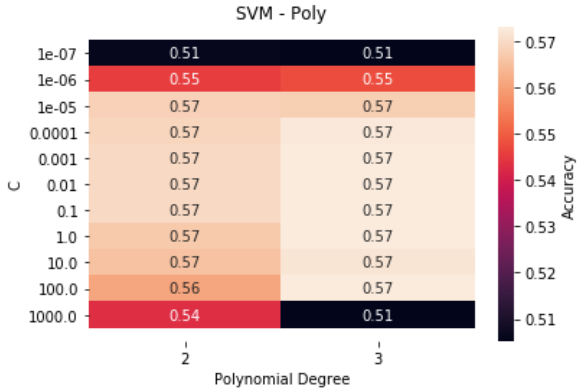
A.3.2



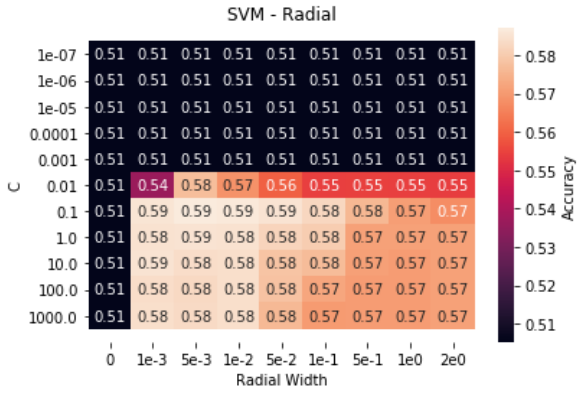
A.3.3



A.3.4


SVM - Poly
C
1e-07 0.51 0.51
1e-06 0.55 0.55
1e-05 0.57 0.57
0.0001 0.57 0.57
0.001 0.57 0.57
0.01 0.57 0.57
0.1 0.57 0.57
1.0 0.57 0.57
10.0 0.57 0.57
100.0 0.56 0.57
1000.0 0.54 0.51
2
3
Polynomial Degree
Accuracy
0.57
0.56
0.55
0.54
0.53
0.52
0.51

A.3.5


SVM - Radial
C
1e-07 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51
1e-06 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51
1e-05 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51
0.0001 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51
0.001 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51 0.51
0.01 0.51 0.54 0.58 0.57 0.56 0.55 0.55 0.55 0.55
0.1 0.51 0.59 0.59 0.59 0.59 0.58 0.58 0.57 0.57
1.0 0.51 0.58 0.59 0.58 0.58 0.58 0.57 0.57 0.57
10.0 0.51 0.59 0.58 0.58 0.58 0.58 0.57 0.57 0.57
100.0 0.51 0.58 0.58 0.58 0.58 0.57 0.57 0.57 0.57
1000.0 0.51 0.58 0.58 0.58 0.58 0.57 0.57 0.57 0.57
0
1e-3
5e-3
1e-2
5e-2
1e-1
5e-1
1e0
2e0
Radial Width
Accuracy
0.58
0.57
0.56
0.55
0.54
0.53
0.52
0.51