Miniproject 2: Classification of Textual Data

Nadia Blostein 260804876 **Vivian Eberle 260745995** **Christopher Scarvelis 260689983**

Abstract

The purpose of this project was to examine the effects of hyperparameter tuning on six different classification models: logistic regression, decision tree, support vector machine (SVM), AdaBoost, random forest and kernel SVM. The performance of these models was tested on a multiclass classification task and a binary classification task, using the 20 newsgroups dataset and the IMDb dataset respectively. After examining different combinations of hyperparameters, we found that the logistic regression performed best on the IMDb dataset with a test accuracy of 88.123% (Sag solver, L2 penalty, regularization parameter C of 2). For the 20 newsgroups dataset, we found that the SVM had the best performance, with a test accuracy of 69.60% (L2 penalty, squared hinge loss, C of 0.5). Based on a weighted average between the best models for each dataset, the overall performance of our models was 82.45%.

1 Introduction

In this project, we compared the accuracy of different classification methods on text data using two different datasets. The 20 newsgroups dataset has been widely used within the context of multiclass text classification, while the IMDb movie reviews dataset is used as a common test case for binary text classification. We leveraged these datasets in order to study the effects of hyperparameter tuning on the performance of six different classification models. We found that linear SVM achieved the best validation accuracy on the 20 newsgroups dataset, whereas logistic regression performed best on the IMDb dataset. In general, methods based on decision trees (including AdaBoost and random forests) underperformed linear classifiers on both datasets.

2 Related work

Within the context of multiclass and binary text classification, certain techniques can be used to facilitate the classification task at hand. For instance, bag-of-words models and term-frequency times inverse document-frequency (TF-IDF) transformers are frequently implemented to pre-process text-data and perform feature selection. [1] [2]. One can obtain a vectorized word count for each instance in a text dataset via bag-of-words representation. These values are then normalized by the total number of words per document and by the frequency of each word's occurrence in the corpus of documents via TF-IDF [1]. Regarding different classifiers' accuracy, Onan (2016) found that when using term-frequency times inverse sentence-frequency (TF-ISF) – a technique similar to TF-IDF – the accuracy for text classification was greatest with Random Forests, at 89.83%, followed by Naïve Bayes (88.79%), Logistic Regression (87.13%) and SVM (86.21%) [10]. With binary text classification using the IMDb dataset, Maas (2011) produced an 88.23% accuracy, having used a normalized bag of words text preprocessing and a linear support vector machine as a classifier [9]. Bouazizi (2016) compared the accuracy of binary and multiclass text classification, fining that Random Forests had a greater accuracy when classifying text in binary categories (87.5%) compared to multiclass categories (56.9%) with seven categories [5].

3 Datasets and setup

3.1 20 newsgroups

The 20 newsgroups dataset can be accessed through `sklearn.datasets`. This dataset is comprised of news articles from 20 different news categories. Each instance is an article in the form of a text file with a total of 18,846 instances.

The dataset is split into train and test subsets containing 11,314 and 7,532 instances, respectively. After training our models on the test instances, our goal was to correctly classify articles from the test subset. The distribution of the outputs was relatively even across categories, reducing any potential category biases. The features used to train our classifiers were comprised of the preprocessed words of the train dataset.

3.2 IMDb reviews

The IMDb reviews dataset can be accessed through `http://ai.stanford.edu/~amaas/data/sentiment/`. This dataset contains movie reviews in the form of text files, labelled as either positive (rating ≥ 7) or negative (rating ≤ 4). We used the labelled train and test data, each subset containing 25,000 instances. The instances were distributed evenly across positive and negative reviews for the train and test data. Our goal with this dataset was to classify correctly the reviews from the test set as either positive or negative. Negative reviews were labelled as 0 and positive reviews were labelled as 1.

3.3 Preprocessing

To process both datasets, we used a count vectorizer where we removed a list of stopwords, kept the default lowercase parameter to "True", and ensured that the words are formatted without numbers or any other character that is not a letter by initializing the token_pattern parameter. As such, the features for both datasets are the tokenized words, and the count vectorizer outputs a vector with the amount of times a given word appears in each instance (text file). For the 20 newsgroup dataset, this results in a total of 67,647 features, and 73,319 features for the IMDb dataset, meaning that those are the number of unique words in each entire dataset after the processing. We also transformed the data into a TF-IDF vector for normalization, to minimize the impact of features that occur very frequently across the respective datasets.

4 Proposed approach

In this section, we will briefly outline the models tested in this project and describe our cross-validation procedure. Much of the background information presented here is found in [3] and [8] as well as in the class slides, but we will cite the papers in which the models were originally published where possible.

4.1 Logistic regression

The first model we test is a logistic regression classifier. Given a vector of features $x \in \mathbb{R}^d$ and a vector of weights $w \in \mathbb{R}^d$, a binary logistic classifier predicts the label associated with x by computing the following function:

$$f_w(x) = \frac{1}{1 + e^{-w^T x}}$$

This classifier induces a linear decision boundary defined by the hyperplane $w^T x = 0$. The optimal weights w are computed by minimizing the *cross-entropy loss*. The multiclass case (with C classes) is conceptually similar, but the predictor function is the *softmax function*. This model may be regularized by adding a penalty on the norm of the weight vector w. In particular, adding an L_2 penalty of the form $R(w) = \frac{\lambda}{2}||w||_2^2$ encourages the components of w to be close to zero, whereas an L_1 penalty $R(w) = \lambda||w||_1$ favors weight vectors w with many zero components and hence enforces *sparsity* in the trained model.

4.2 Decision tree

Decision trees are an example of an adaptive basis function model. The basic approach is to divide the feature space into finitely many regions and learn a very simple (typically constant) function in each region, leading to a model of the following form:

$$f(x) = \sum_{k=1}^{K} w_k \mathbb{I}(x \in \mathbb{R}^k)$$

The regions are learned via a recursive procedure that in each step splits each region into two subregions by greedily selecting a test that minimizes a cost function. Typical examples of cost functions for classification are the entropy cost and the Gini index, both of which favour pure nodes. The primary disadvantage of decision trees is that they easily overfit unless techniques such as pruning or bagging are employed to reduce their variance.

4.3 Support vector machine

A support vector machine (SVM) is another example of a linear classifier. Unlike logistic regression, an SVM seeks a linear decision boundary that achieves the maximum *margin* in feature space. This amounts to finding a weight-bias pair (w, w_0) minimizing the function $||w||_2^2$ subject to the inequalities $y_i(w^T x_i + w_0) \geq 1$ for all (x_i, y_i) in the training set. This is known as the hard-margin SVM objective; it is a convex minimization problem that may be solved to global optimality via first-order methods. Oftentimes, the training data one encounters is not linearly separable. In this case, one may relax the hard margin constraints to allow for training points to be misclassified by the optimal SVM but penalize points that are misclassified or within the margin. Alternatively, one may use a *kernel* SVM. (See [4] for a discussion of this algorithm and SVM in general.) This method non-linearly embeds the features in a typically higher-dimensional space and then learns a max-margin linear classifier in this space.

4.4 AdaBoost

AdaBoost is an *ensemble method* that was introduced in the landmark paper of Freund and Schapire [7]. This algorithm iteratively builds a strong learner from a sequence of weak learners (i.e. high-bias, low-complexity models) such as decision stumps. In particular, AdaBoost builds a function of the following form:

$$f(x) = \sum_{t=1}^{T} w_t \phi(x; v_t)$$

where each $\phi(x; v_t)$ is a different weak learner. By minimizing an exponential loss, AdaBoost places a higher weight on misclassified points throughout the training procedure and hence favours weak learners that correctly classify points that have been misclassified in earlier stages. This algorithm has been found empirically to be resistant to overfitting, making it an attractive choice as an off-the-shelf classifier.

4.5 Random forest

Random forests are another ensemble method for classification. Introduced by Breiman in [6], random forests implement a variant of bagging for decision trees. As noted above, decision trees have a strong tendency to overfit their training data. To avoid this outcome, bagging subsamples the training set with replacement to obtain a collection of B bootstrap samples. A decision tree is trained on each of these samples, and their predictions are aggregated at test time using a voting procedure. Provided that the predictions of each tree are uncorrelated, the resulting classifier will exhibit lower variance than any of the base trees. To decrease the correlation of the base trees, random forests select a random subset of features as candidates for each split in the training procedure for the trees.

4.6 Cross-validation procedure

We use scikit-learn [11] to cross-validate and test the models used in this project. We build a Pipeline object that includes the count vectorizer and TF-IDF transformer described above. For each model, we select a subset of hyperparameters to test during cross-validation and otherwise rely on the default hyperparameters for each model in scikit-learn. In certain cases, we carry out 5-fold cross-validation using the built-in RandomizedSearchCV class to reduce our computational load. Oftentimes, the number of hyperparameter combinations that we test is under 10; in this case, RandomizedSearchCV with its default n_iters parameter value simply reduces to grid search over the parameter grid. After carrying out cross-validation, we choose the combination of hyperparameters with the best validation score and evaluate the accuracy of each model with its best hyperparameters on the test set to obtain our final results.

5 Results

All relevant figures for this section are in the appendix. Figure 1 in Appendix 0 provides the validation results for each model with the best hyperparameters. Broadly speaking, our goal here is to explain our cross-validation approches where appropriate and to outline our results.

5.1 Logistic Regression

5.1.1 20 newsgroups dataset

We compared the performances of the saga, sag, lbfgs and newton-cg solvers with no regularization; all achieved fairly good accuracy. Adding a L_2 penalty, the accuracy remained similar but the fit times decreased. Since the saga solver

can handle 3 types of penalties [11] and performed quite well with an L_2 penalty, we examined its performance across all penalties. The ElasticNet penalty achieved an accuracy of 0 and had a fast mean fit time, implying that the procedure likely failed. Finally, we compared the sag and saga solvers with an L_2 norm penalty and no penalty. Given that the mean fit time was slightly lower for the sag solver across all tested C values, we decided to run our model with the sag solver and a L_2 norm penalty with a regularization parameter $C = 6$.

5.1.2 IMDb dataset

For the IMDb dataset (which has binary targets), the training procedure for logistic regression failed when we used no penalty. Furthermore, the saga solver failed to reach optimality for any penalty. Amongst the three other solvers, we selected the sag solver using an L_2 penalty. Its mean accuracy was slightly superior and it also had the lowest mean fit time. Stronger regularization was typically required relative to the 20 newsgroups dataset.

5.2 Decision Tree

5.2.1 20 newsgroups dataset

For both models (using either the Gini or entropy criteria), the optimal minimal number of samples at a leaf node was 1. Because randomized search CV is not exhaustive, our estimates for min sample split varied between 2 and 3 every time we ran this CV. We therefore decided to compare both of these values for both split quality criteria. The Gini criterion significantly outperformed the entropy criterion for both values of min samples to split. Until a tree depth of 1000, the mean fit time was lower with a min samples to split value of 2. At a depth of 1000, the mean fit times of the models with a min sample to split value of 2 and 3 converged. However, for all four models, our test accuracy was low, with a range of 32.6% (Entropy criterion, min sample split value of 2) to 44.14% (Gini criterion, min sample split value of 2).

5.2.2 IMDb dataset

Based on an initial run, we decided to set the min sample per leaf at 1 for both the Gini and Entropy criterion. Given that our results for min samples required to split were fairly consistent across several runs, we set that value at 3 for the Gini criterion and at 2 for the entropy criterion. We found that the entropy criterion generally outperformed the Gini criterion with respect to accuracy. Interestingly, there is no clear relationship between max tree depth and accuracy for either criterion on this dataset.

5.3 Support Vector Machine (SVM)

5.3.1 20 newsgroups dataset

Scikit-learn's LinearSVC can handle either L_1 and L_2 penalties and a hinge or squared hinge loss function. When comparing through models via Randomized Search CV (Grid Search took too long to optimize), the L_2 model that implemented a squared hinge loss function had the highest CV score (76%) using relatively strong regularization ($C = 0.5$). In general, we found that the L_1 penalty performs relatively poorly, while the L_2 penalty with hinge and squared hinge losses perform comparably.

5.3.2 IMDb dataset

For the IMDb dataset, we considered only the the hinge loss with L_2 penalty. With these settings, the SVM model yielded good accuracy for all tested values of C. However, validation accuracy was monotonically decreasing in C and so we chose $C = 1$ as our optimal hyperparameter for this model.

5.4 AdaBoost

5.5 20 newsgroups dataset

On this dataset, AdaBoost with depth-4 base learners exhibits somewhat better performance with many base learners provided that the learning rate is small (on the order of 0.5). For larger learning rates, we observe steadily worsening performance as the number of base learners is increased. In all cases, the results are relatively poor, with validation accuracy no better than 0.48 for any hyperparameter values. The results were broadly similar for depth-2 base learners, but in this case the validation accuracy improved at a somewhat higher rate as the number of base learners was increased for a learning rate of 0.5.

5.5.1 IMDb dataset

We found that increasing the number of base learners generally decreased the validation performance within the range we tested, which indicates that AdaBoost begins to overfit as more base learners are added. Lower learning rates performed uniformly better than higher rates, with particularly poor results for a learning rate of 2, where AdaBoost performed little better than chance. This behaviour existed both for base lerners of depth 2 and 4, with the results for depth-4 trees being somewhat worse. This also points to deeper trees overfitting the training data.

5.6 Random forest

5.6.1 20 newsgroups dataset

As with vanilla decision trees, random forests with trees built using the Gini criterion yielded substantially better performance than trees built with the entropy criterion. In both cases, performance is slightly better when the minimum number of samples required to split is 3, and performance improves slightly as the number of base trees increases.

5.6.2 IMDb dataset

On this dataset, the entropy criterion slightly outperformed the Gini criterion (though the differences are marginal). Furthermore, increasing the number of base trees increases performance slightly for both criteria and for all values of the minimum number of samples required to split.

5.7 Kernel SVM

5.8 20 newsgroups dataset

As an additional experiment, we also investigated the use of a kernel SVM classifier (implemented by scikit-learn's SVC class). We tested the widely-used RBF kernel, along with polynomial kernels of various degrees and a linear kernel (which should be comparable to the LinearSVC classifier). We found that the RBF kernel performs fairly well for large enough values of C on this dataset, but accuracy is poor for small values of C. For the degree-2 polynomial kernel, accuracy peaks at $C = 1$ and declines for larger and smaller values; this kernel performs comparably to RBF. Training appears to fail for the degree-3 kernel, as validation accuracy is extremely poor in this case.

5.9 IMDb dataset

For this dataset, training of the kernel SVM classifiers appeared to fail for all kernels but the linear kernel. Runtimes were also very long, which may be explained by the fact that kernel SVMs have time complexity that is quadratic in the number of training samples. As the IMDb dataset is quite large, training is very long as well in this case. This issue is not as severe for LinearSVC, as this class uses a solver that scales better to large training sets. (See [11] for details.)

6 Discussion and conclusion

The table in Figure 1 (see Appendix 0) records the mean validation accuracy of each model (with their best hyperparameters) on both datasets. In general, we find that decision trees perform quite poorly on both datasets, though ensemble models based on decision trees (such as AdaBoost and random forests) achieve somewhat better accuracy. Linear classifiers such as logistic regression and linear SVM perform best on both datasets. We achieve test accuracy similar to the TA's baseline on both datasets (our results are slightly better for 20 newsgroups and slightly worse for IMDb.) Future work may consider exploring the effect of pre-processing the datasets with word embeddings such as Stanford's GloVe on the classification accuracy of these models.

7 Statement of contributions

Vivian prepared the datasets, Chris coded the classifiers and Nadia tested the classifiers with different hyperparameters. Each member did their respective part of the write-up.

References

[1] Charu C. Aggarwal and Cheng Xiang Zhai. "A survey of text classification algorithms". English (US). In: *Mining Text Data*. Vol. 9781461432234. Springer US, Aug. 2012, pp. 163–222. ISBN: 1461432227. DOI: 10.1007/978-1-4614-3223-4_6.

[2] Leila et al. Arras. "What is relevant in a text document? An interpretable machine learning approach". In: *PLoS ONE 12(8): e0181142* (2017). DOI: https://doi.org/10.1371/journal.pone.0181142.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *COLT 92* (1992), pp. 144–152. DOI: https://doi.org/10.1145/130385.130401.

[5] M. Bouazizi and T. Ohtsuki. "Sentiment analysis: From binary to multi-class classification: A pattern-based approach for multi-class sentiment analysis in Twitter". In: *IEEE International Conference on Communications (ICC)* (2016), pp. 1–6. DOI: 10.1109/ICC.2016.7511392.

[6] Leo Breiman. "Random Forests". In: *Machine Learning* (2001), pp. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

[7] Yoav Freund and Robert E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* (1997), pp. 119–139. DOI: https://doi.org/10.1006/jcss.1997.1504.

[8] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. New York, NY: Springer, 2009.

[9] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: https://www.aclweb.org/anthology/P11-1015.

[10] H. Bulut Onan S. Korukoglu. "Ensemble of keyword extraction methods and classifiers in text classification". In: *Expert Systems with Applications* (2016), pp. 232–247. DOI: https://doi.org/10.1016/j.eswa.2016.03.045.

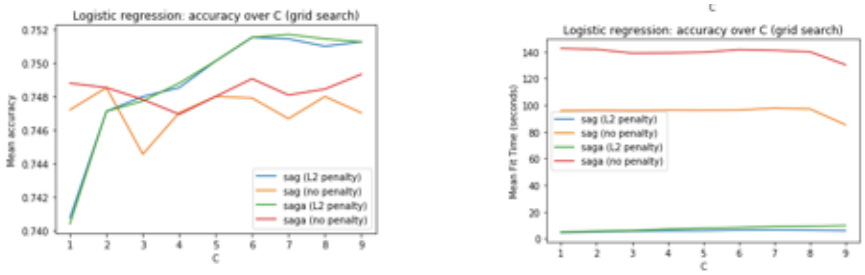
[11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

8 Appendix 0: Table of results

		Newsgroups	IMDb	Best model test accuracy, per dataset
Logistic regression	Validation Accuracy:	75.15%	89.46%	IMDb: 88.128 %
	Best hyperparams:	Sag solver, L2 penalty, C = 6	Sag solver, L2 penalty, C = 2	
Decision tree	Validation Accuracy:	46.76%	71.80%	
	Best hyperparams:	Gini criterion, min sample split = 2, min sample leaf = 1, max depth = 750	Entropy criterion, min sample split = 2, min sample leaf = 1, max depth = 500	
SVM	Validation Accuracy:	76.05%	89.15%	Newsgroups: 69.60 %
	Best hyperparams:	L2 penalty, squared hinge loss, C = 0.5	L2 penalty, hinge loss, C = 1	
AdaBoost	Validation Accuracy:	48.94%	84.69%	
	Best hyperparams:	Tree depth = 2, learning = rate 0.5, best n estimators = 1000	Tree depth = 2, learning rate = 0.5, best n estimators = 200	
Random forest	Validation Accuracy:	68.00%	86.34%	
	Best hyperparams:	Gini criterion, min sample split = 3, n estimators = 500, min samples leaf = 1	Entropy criterion, min sample split = 3, n estimators = 500, min samples leaf = 1	
Kernel SVM	Validation Accuracy:	73.87%	87.82%	
	Best hyperparams:	Linear kernel, C = 1	Linear kernel, C = 1	
Overall				**82.45%**

Figure 1: Best hyperparameters and validation accuracy across models and datasets

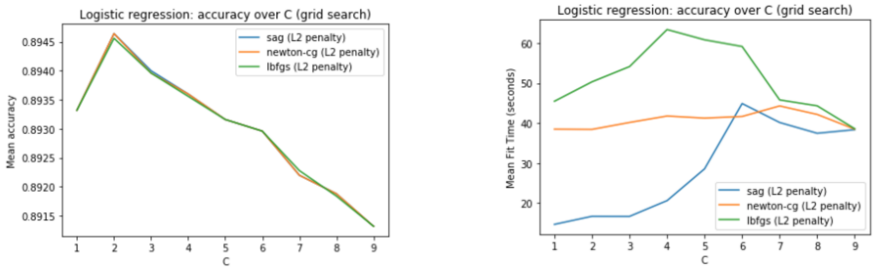
9 Appendix 1: Logistic Regression Graphs



(a) Mean accuracy across C

(b) Mean fit time across C

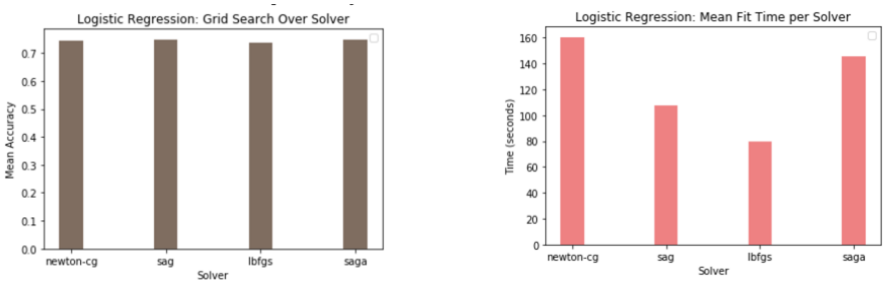
Figure 2: Results of logistic regression on 20 news groups



(a) Mean accuracy across C

(b) Mean fit time across C

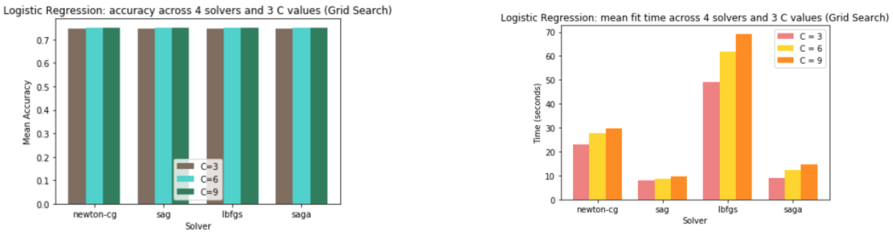
Figure 3: Results of logistic regression on IMBD dataset



(a) Mean accuracy per solver

(b) Mean fit time per solver

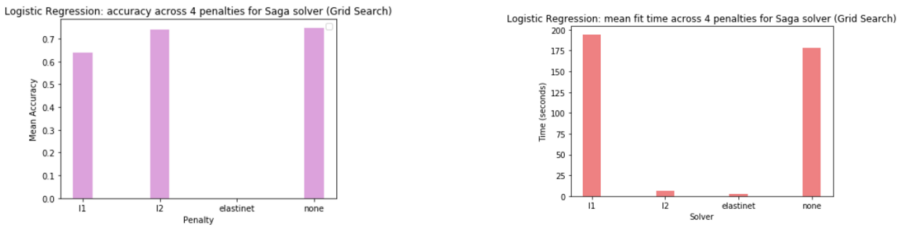
Figure 4: Results of logistic regression with no regularization across solvers (20 newsgroups dataset)



(a) Mean accuracy per solver, per C value

(b) Mean fit time per solver, per C value

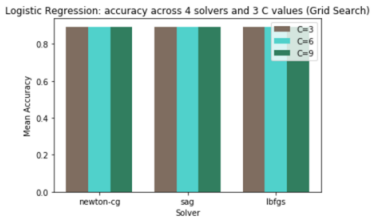
Figure 5: Results of logistic regression with L2 across solvers (20 newsgroups dataset)



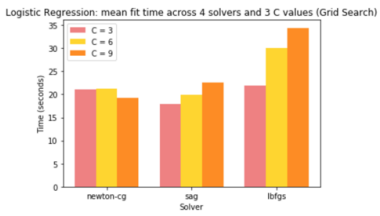
(a) Mean accuracy per penalty

(b) Mean fit time per penalty

Figure 6: Results of logistic regression with Saga solver across penalties (20 newsgroups dataset)



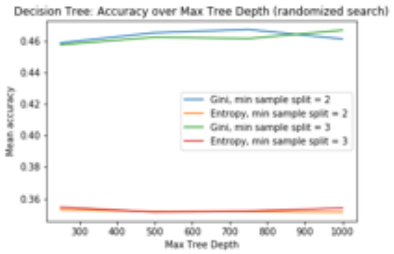
(a) Mean accuracy per solver, per C-value



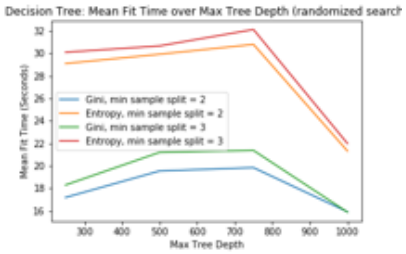
(b) Mean fit time per solver, per C-value

Figure 7: Results of logistic regression with L2 across solvers (IMDb dataset)

10 Appendix 2: Decision Tree Graphs



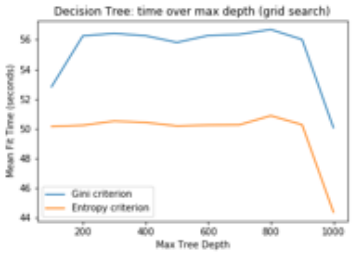
(a) Mean accuracy across max tree depth



(b) Mean fit time across max tree depth

Figure 8: Results of decision tree on 20 newsgroups dataset

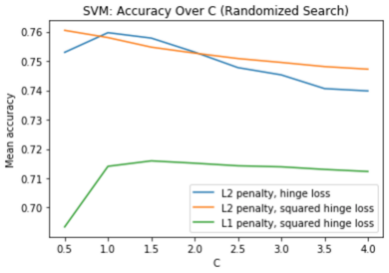
(a) Mean accuracy across max tree depth



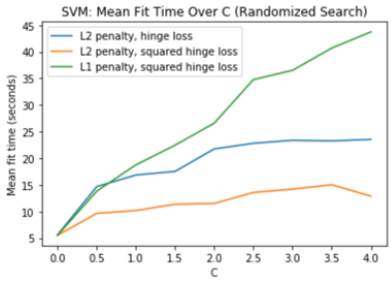
(b) Mean fit time across max tree depth

Figure 9: Results of decision tree on IMDb dataset

11 Appendix 3: SVM Graphs

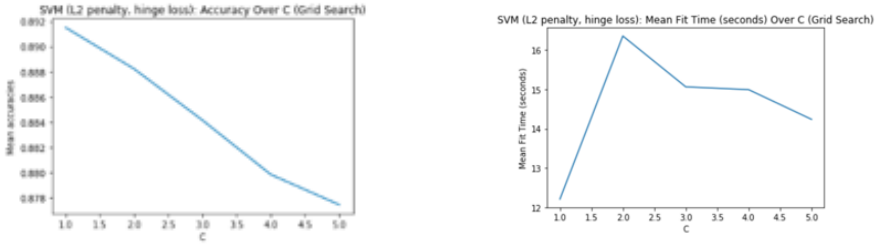


(a) Mean accuracy across C



(b) Mean fit time across C

Figure 10: Results of SVM on 20 newsgroups dataset

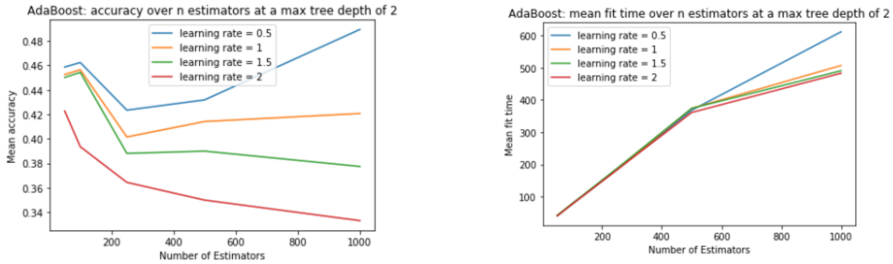


(a) Mean accuracy across C

(b) Mean fit time across C

Figure 11: Results of SVM on IMDb dataset

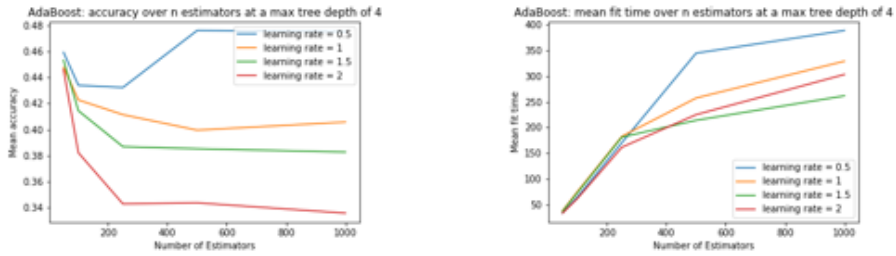
12 Appendix 4: AdaBoost Graphs



(a) Mean accuracy across n estimators

(b) Mean fit time across n estimators

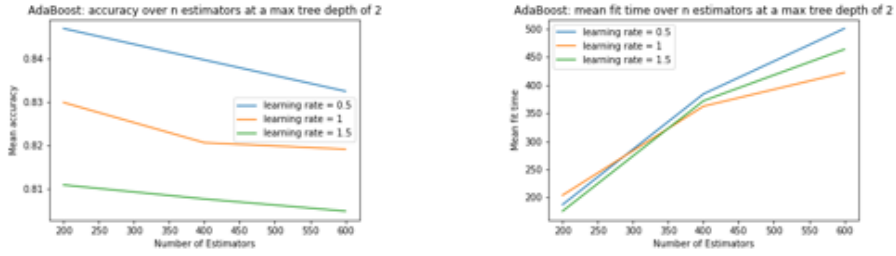
Figure 12: Results of AdaBoost with depth of 2 on 20 newsgroups dataset



(a) Mean accuracy across n estimators

(b) Mean fit time across n estimators

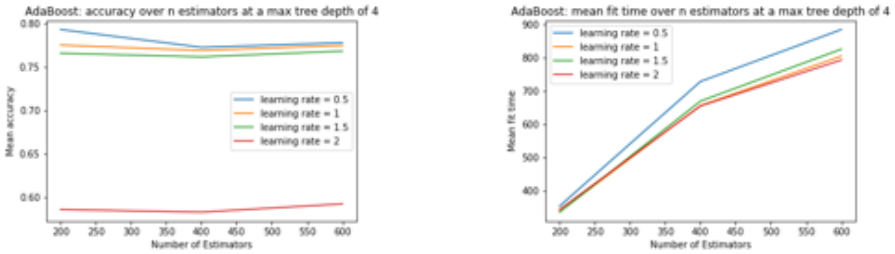
Figure 13: Results of AdaBoost with depth of 4 on 20 newsgroups dataset



(a) Mean accuracy across n estimators

(b) Mean fit time across n estimators

Figure 14: Results of AdaBoost with depth of 2 on IMDb dataset

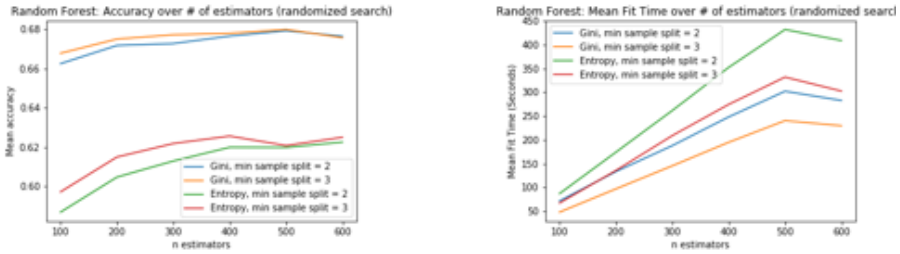


(a) Mean accuracy across n estimators

(b) Mean fit time across n estimators

Figure 15: Results of AdaBoost with depth of 4 on IMDb dataset

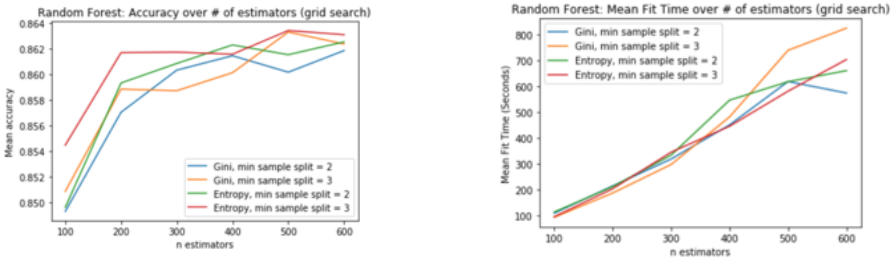
13 Appendix 5: Random Forest Graphs



(a) Mean accuracy across n estimators

(b) Mean fit time across n estimators

Figure 16: Results of Random Forest on the 20 newsgroups dataset

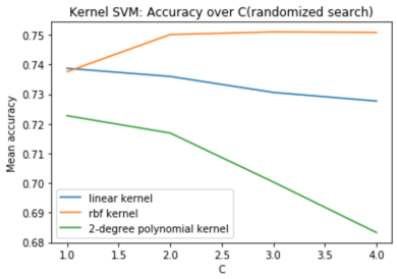


(a) Mean accuracy across n estimators

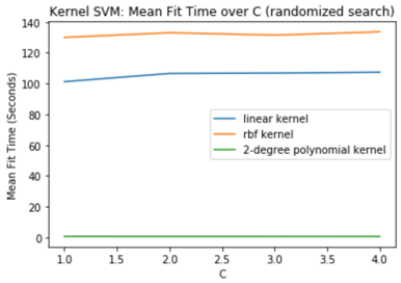
(b) Mean fit time across n estimators

Figure 17: Results of Random Forest on the IMDb dataset

14 Appendix 6: Kernel SVM Graphs

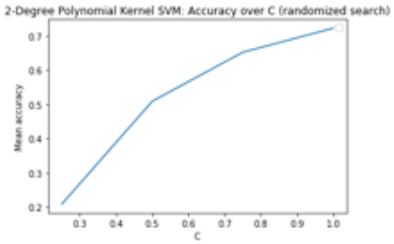


(a) Mean accuracy across C

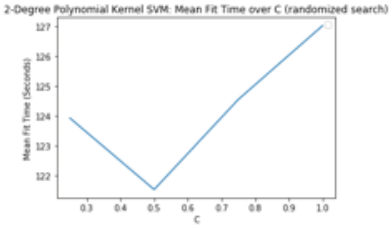


(b) Mean fit time across C

Figure 18: Results of Kernel SVM on range of 1 to 4 for 20 newsgroups dataset

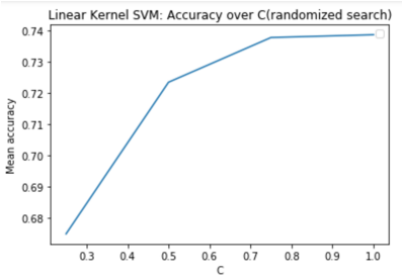


(a) Mean accuracy across C

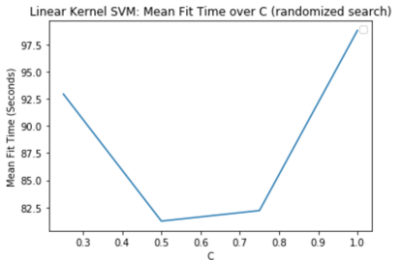


(b) Mean fit time across C

Figure 19: Results of 2-degree polynomial kernel SVM on range of 0 (exclusive) to 1 (inclusive) for 20 newsgroups dataset

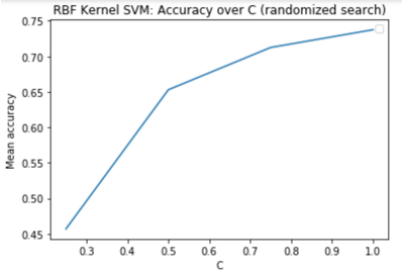


(a) Mean accuracy across C

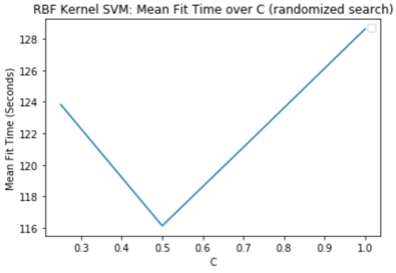


(b) Mean fit time across C

Figure 20: Results of linear kernel SVM on range of 0 (exclusive) to 1 (inclusive) for 20 newsgroups dataset

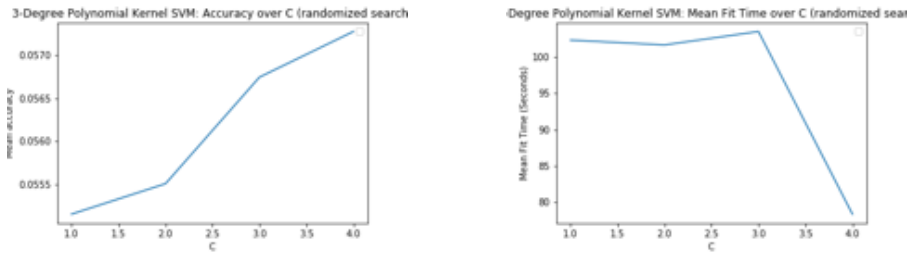


(a) Mean accuracy across C



(b) Mean fit time across C

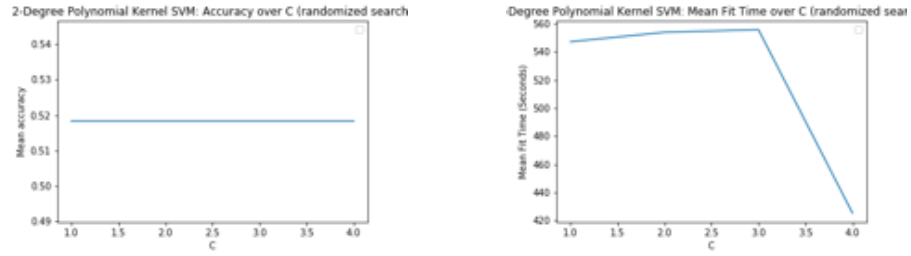
Figure 21: Results of RBF kernel SVM on range of 0 (exclusive) to 1 (inclusive) for 20 newsgroups dataset



(a) Mean accuracy across C

(b) Mean fit time across C

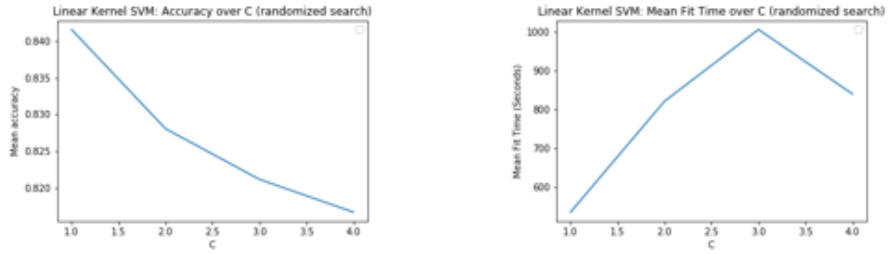
Figure 22: Results of 3-degree polynomial kernel SVM on range of 1 to 4 for 20 newsgroups dataset



(a) Mean accuracy across C

(b) Mean fit time across C

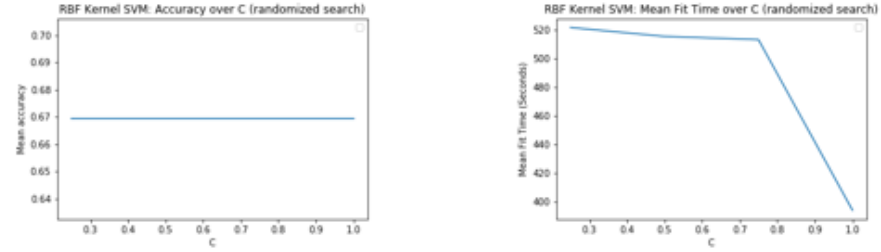
Figure 23: Results of 2-degree polynomial kernel SVM on range of 1 to 4 for IMDb dataset



(a) Mean accuracy across C

(b) Mean fit time across C

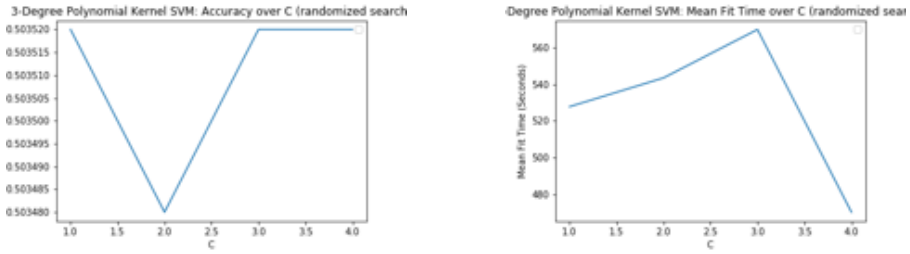
Figure 24: Results of linear kernel SVM on range of 1 to 4 for IMDb dataset



(a) Mean accuracy across C

(b) Mean fit time across C

Figure 25: Results of RBF kernel SVM on range of 1 to 4 for IMDb dataset



(a) Mean accuracy across C

(b) Mean fit time across C

Figure 26: Results of 3-degree polynomial kernel SVM on range of 1 to 4 for IMDb dataset