Analysis and Comparison of Popular Machine Learning Algorithms on Contextual Data

Andrew Cheng Chris Rafiaa Stephanie Li

Abstract

Many machine learning algorithms have been developed in recent history. We will explore the performance of some of the most common models in this paper given a categorical or a binary classification problem on text files. These models include Adaboost, linear SVC, linear regression, radial basis function SVM, random forest, decision tree, and BERT. Our results show the effects of regularization, resampling methods such as bagging (bootstrap aggregating) and 5-fold cross-validation as well as boosting on model accuracy. We also examine the effects of these strategies on bias-variance tradeoff to determine the best models for each algorithm and data set. Our highest test accuracies were achieved using BERT: 72.40% on 20 Newsgroups and 94.15% on IMDb.

Introduction

We first present the background and motivations of each algorithm. Then we introduce our approach in processing the data and finding the optimal hyperparameters. Finally, we report the best hyperparameters for each model and accuracies. The goal of this project is to be accustomed to the empirical nature of model selection. In particular we explore the effects of hypertuning as well comparing the complexities of each model. In addition, all our figures are included in the appendix and referred to as necessary.

Kernel Trick
Linear classifiers tend to underperform on nonlinear data. The kernel trick allows linear classifiers to learn nonlinear trends by projecting the input features into higher dimensions via a *kernel function*. By doing so, the data in the transformed space is likely to be linearly separable, enabling our linear classifiers to perform well in the transformed space. One such kernel function is the Radial Basis Function (RBF), $K(x_i, x_j) = e^{-\gamma(x_i - x_j^2)}$, which sums infinitely many polynomial kernels of different degrees, guaranteeing our data to be linearly separable. [1]

Logistic regression
Logistic regression falls under the linear discriminative learning framework as it learns optimal linear decision boundaries to predict the class of a new data point. In other words, given N observations and D features, logistic regression assumes the linearity of independent variables and log-odds:

$$f_w(x_i) = \log(\frac{y_i}{1-y_i}) = w_0 + \sum_{j=1}^{D} w_j x_{ij}$$

and our goal is to find a set of weights $\mathbf{w} = \{w_0 \cdots w_D\}^T$ that maximize the log-likelihood:

$$L(\mathbf{w}) = \sum_{i=1}^{N} y_i \log(\sigma_{\mathbf{w}}(x_i)) + (1-y_i)\log(1-\sigma_{\mathbf{w}}(x_i)).$$

where $\sigma_{\mathbf{w}}(x) = \frac{1}{1+\exp(-\mathbf{w}^T x)}$ is the sigmoid function (the inverse of log-odds). Equivalently, we can find weights that minimize the cross entropy loss function, $-L(\mathbf{w})$, a convex function. As the L2 norm, $||\mathbf{w}||_2^1$, which induces sparsity in models, is also convex, we have that the optimization problem,

$$argmin_{\mathbf{w}}(-L(\mathbf{w}) + \lambda||\mathbf{w}||_2^2) \tag{1}$$

is convex for any $\lambda \in \mathcal{R}$. This means that stochastic gradient descent can find the weights to minimize (1).

Support Vector Machines (SVM)
SVM also belongs to the family of linear classifiers. Its learning is driven by seeking the hyperplane with maximum margin with respect to the different classes in our dataset *D*. Let $\mathcal{H}_{w,b} = \{x|\mathbf{w}^T x + b = 0\}$ be the set of all possible

[1] In general, n points are linearly separable in n-1 dimensional space so it trivially holds that n points will be linearly separable in infinite dimensional space.

hyperplanes for weights **w** and bias b. If the data we are working with has few dimensions, the dataset is unlikely to be linearly separable and no such hyperplane can be found.

To remedy this, we introduce the *slack variables* which yields the following optimization problem:

$$min_{\mathbf{w},b}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\eta_i \quad s.t \quad \forall i\ y_i(\mathbf{w}^T x_i + b) \geq 1 - \eta_i \tag{2}$$

where $\eta_i \geq 0$ and $C \in [0, \infty)$. As we increase the penalty parameter C, SVM will find the hyperplane that classifies more training points correctly thereby increasing the risk of overfitting. Optimizing (3) is equivalent to optimizing a convex loss function.

$$J(\mathbf{w}) = min_{w,b}\left(\underbrace{\mathbf{w}^T\mathbf{w}}_{L2-regularization} + C\sum_{i=1}^{n}\underbrace{max(1 - y_i(\mathbf{w}^T x_i + b), 0)}_{Hinge\ Loss}\right) \tag{3}$$

Since L2-regularization and hinge loss are both convex, (3) is convex as well and gradient descent methods will find the global optimum.

Radial Basis Function Kernelized SVM
Let ϕ be the RBF transformation and K be a DxD matrix (D is the number of features in our data set) such that $K_{ij} = \phi(x_i)^T\phi(x_j)$. If we use the RBF kernel SVM, we can use the fact that the primal SVM optimization problem (3) has the dual form:

$$min_{\alpha_1,\cdots,\alpha_n}\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K_{ij} - \sum_{i=1}^{n}{}_i \quad s.t \quad 0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_{i=1}^{n}\alpha_i y_i = 0$$

where $\alpha_i \in \mathbf{R}$, $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \phi(x_i)$ and $h(x) = sign\left(\sum_{i=1}^{n}\alpha_i y_i k(x_i, x) + b\right)$ (Kernels, 2020). So essentially we solve the dual problem and we use h(x) as our classification function. In this paper, we explore the effects of the slack variables and RBF kernel when applied to SVM.

Boosting Algorithms:
Consider the hypothesis class $\mathcal{H}$, where each classifier $h \in \mathcal{H}$ is a weak learner (high bias and training error). The main objective of boosting algorithms is to ensemble many weak learners in order to construct a strong learner (a classifier that is arbitrarily well-correlated with the true classification). This ensemble can be expressed as a linear combination of the weak learners: $H_M(x) = \sum_{k=1}^{M}\alpha_k h_k(x)$. This ensemble is found iteratively via gradient descent in functional space, where the inner product of the space is defined as $< h, g >= \sum_{i=1}^{n} h(x_i)g(x_i)$ as opposed to $< h, g >= \int_x h(x)g(x)dx)$, since we only have the training data.

Adaboost
Adaboost is a binary classification boosting algorithm that utilizes an exponential loss function $\mathcal{L}(H_M) = \sum_{i=1}^{n} e^{-y_i H_M(x_i)}$ in order to converge extremely fast. In the adaboost setting, the next optimal weak learner added to our ensemble is solved by the following optimization problem:

$$h(x_i) = argmin_{h\in\mathcal{H}}\sum_{i:h(x_i)\neq y_i} w_i z \tag{4}$$

where $w_i = \frac{1}{F}e^{-y_i H_M(x_i)}$ and $F = \sum_{i=1}^{n} e^{-y_i H(x_i)}$ is a normalization factor to ensure $\sum_i^n w_i = 1$. The optimal stepsize can be found in closed form: $\alpha = \frac{1}{2}\ln\frac{1-\epsilon}{\epsilon}$, which is very powerful as each iteration utilizes the optimal step size enabling exceptionally fast convergence. The Adaboost classifier we implemented uses an aggregation of differently weighted tree stumps (1-level decision trees).

Bidirectional Encoder Representations from Transformers (BERT)
BERT is a state-of-the-art Natural Language Processing (NLP) algorithm (Devlin, et al. 2018) motivated by the lack of textual training examples available for deep learning models. Deep learning based NLP models are known to require training examples on the magnitude of millions, or even billions, to see major improvements. However, we are often restricted to only thousands of training examples for specific NLP tasks. BERT is based on the *Transformer model architecture*, which directly models all words in a sentence. The key innovation of BERT is that it is *bidirectionally trained*. This means that before feeding sequences into BERT, the algorithm randomly masks 15% of the words in the sequence. The model then tries to predict the original masked word by reading the entire sequence of words at once.

This process is known as *Masked Language Modelling* (MLM). In short, BERT learns a *bidirectional* and *context-based* representation of sentences, which contrasts with *context-free* fixed-length NLP models that output a real-valued vector for each word and do not take into account how the words are ordered in the sequence. The loss function of BERT only considers whether the masked words are correctly predicted and the prediction of the non-masked words are irrelevant. In addition, BERT is a model pretrained on two NLP tasks: MLM and Next Sentence Prediction using the BooksCorpus and Wikipedia. BERT's architecture differs in terms of number of layers and hidden units depending on which version is used. We used BERT-Base which is a 12-layer neural networks with 768 hidden units and 110 million parameters.

Decision Trees
Decision trees are one of the most commonly used models in machine learning, known for being insensitive to outliers and robust over a wide range of inputs. Rather than having a fixed set of bases, decision trees also allow for adaptive bases where each split is determined greedily. Unlike many other models, decision trees offer an intuitive interpretation of classification because the trained models yield decision-making flow charts. However, decision trees also tend to be unstable and can easily overfit the data. Therefore, trees must always be regularized to obtain more generalizable models. Regularization methods include variations on early stopping and pruning. Common examples of early stopping techniques include limiting tree depth, setting a minimum number data points in a given region, stopping once a certain node impurity is obtained, and terminating when the decrease in impurity reaches a threshold. For the purposes of this paper, we will focus on the effects of pruning on generalizability. There are also multiple methods of calculating node purity, including misclassification error, entropy, and the Gini index. We will also do a brief comparison between Gini and entropy.

Random Forests
Random forests make use of model averaging to obtain more stable and accurate models. This is obtained through bagging (bootstrap-aggregating), where many decision trees are trained on bootstrapped datasets. The concept of bagging relies on the assumption that the individual weak learners are uncorrelated and better than random. However, given that the trees will be sampled from the same dataset, we need to decrease the correlation between trees by sampling a subset of the input features. We will explore the effects of using different numbers of features, where more features are predicted to increase the correlation between the trees, thus decreasing the overall model accuracy. Bootstrapping also allows us to simultaneously train and validate the trees in the forest by using out of bag samples as validation sets. We will also investigate how increasing the correlation between trees impacts the model by training a forest without bootstrapping.

Datasets and Preprocessing

The 20 Newsgroups dataset contains around 18,000 documents, partitioned (nearly uniformly) across 20 different newsgroups. The IMBD dataset comprises of 50,000 highly polarized movie reviews for text analytics. A negative movie review has a rating of 1-4 whereas a positive one has a rating of 7-10.

Each dataset came with two folders - one containing text files for training and another for testing, which we kept these separate for training and testing. We removed the stop words - 'and', 'is', 'the' and, 'of' as well as HTML linebreaks.

We used the *term frequency-inverse document frequency* (tf-idf) as our key preprocessing method. It assigns higher weight to terms that appear often in a particular document, but not in other documents in the corpus. This is a measure of how expressive a term is of the document theme. The following equation assigns the weights of a word *w* for a given document *d*: $tfidf(w, d) = tf \log(\frac{N+1}{N_w+1}) + 1$, where *N* is the number of documents in the training set, N_w is the number of documents in the training set that the words *w* appears in, and *tf* is the number of times that the word *w* appears in *d* (the document you want to transform). Furthermore, we used *n-grams* which changes the range of tokens that are considered as features. In particular, we set the *ngram range* to (1,2) for all models.

Proposed Approach

We used a pipeline that utilizes a GridSearch 5-fold cross-validation (CV) across the chosen hyperparameters through the GridSearchCV() and the make_pipeline() methods in scikit-learn. We chose to incorporate tf-idf into the pipeline to transform the data because it yielded better results than Bag-of-Words. We also ensured that the splitting of the dataset during cross-validation occurs *before* transformations were made, preventing statistical properties of the test set from being leaked into the training set, which would overestimate the test accuracy.
Our pursuit for the optimal parameters is motivated by the bias-variance trade-off. After searching through a grid of hyperparameters, we investigated the model with the best mean CV score. If the standard deviation of its CV score was small (low variance) and over-fitting was not an issue (CV score is approximately equal to the test score) then we either accepted the model or increased the complexity. If variance was high, then regularization methods were used.

For decision trees, we used the minimal cost-complexity pruning regularization strategy, which involved setting the ccp_alpha hyperparameter of the model using the built-in cost_complexity_pruning_path function from scikit-learn. Minimizing overfitting using minimal cost-complexity pruning is preferred to setting limits on the maximum depth or minimum number of leaf samples for early stopping because the process is more intuitive. A tree is fully built, then pruned based on which nodes can be removed in such a way that minimizes the total misclassification rate of the tree. The greedy process is repeated iteratively until there are no more internal nodes with effective alpha values less than the threshold indicated by the alpha hyperparameter. Intuitively, larger alphas make the tree more generalizable as they cause more pruning. In other words, minimal cost-complexity pruning seeks to minimize $R_\alpha(T) = R(T) + \alpha|T|$ where $R(T)$ is the misclassification rate of the tree and $|T|$ is the number of nodes. Furthermore, we define the effective alpha value of a node t to be $\alpha_{eff} = \frac{R(t)-R(T_t)}{|T|-1}$ and we remove the node with the lowest α_{eff} in each iteration until there are no nodes remaining with an α_{eff} less than α.

The training accuracies of the random forest models were determined using out-of-bag (OOB) samples, allowing us to validate and fit the model simultaneously. This bagging method increases overall model accuracy.

To investigate the above, we used classes implemented by scikit-learn including LogisticRegression, AdaBoostClassifier, LinearSVC, DecisionTreeClassifier, and RandomForestClassifier. For logistic regression, we chose to use LogisticRegression rather than SGDClassifier because it exhibited better performance. BERT was used from the ktrain library. Any unspecified parameters were set as scikit-learn's default parameters.

Results

Dataset	**20 Newsgroups**		**IMDb**	
Models	**Test**	**Validation**	**Test**	**Validation**
Adaboost	44.80%	47.35%	84.75%	84.12%
Linear SVC	68.16%	74.31%	90.52%	90.70%
Logistic Regression	66.14%	73.06%	90.20%	90.64%
Radial Basis Function SVM	65.37%	68.02%	90.10%	89.83%
Random Forest	61.25%	65.37%	85.25%	84.55%
Decision Tree	57.00%	43.15%	78.63%	74.66%
BERT	**72.40%**	97.00%	**94.15%**	98.38%

Table 1: Test and validation accuracies for each model and dataset. All validation accuracies were computed using the mean scores 5-fold cross-validation except for those of random forests and BERT, which reports out-of-bag accuracy and one-fold cross validation, respectively.

Logistic Regression
The optimal parameters found are C = 100 for both datasets.

SVM and RBF-SVM
The optimal parameters for kernelized SVM are shown below:

Dataset	**gamma**	**C**
20 Newsgroups	1	10
IMDb	1	10

The unkernelized SVM optimal parameter is C = 1 for both dataset.

Adaboost

We used AdaBoostClassification() with the base estimator set as "DecisionTreeClassifier." The optimal parameters are shown below:

Dataset	**learning rate**	**n estimators**
20 Newsgroups	0.75	150
IMDb	0.75	150

Decision Tree

Our optimal models were found by training many separate models on differnt parameters including the criterion (Gini index vs entropy).

The difference between using Gini and entropy as the splitting criterion was evident in the 20 Newsgroups data set and less pronounced in IMDb. The testing accuracies without any regularization are shown below:

Dataset	**Entropy**	**Gini**
20 Newsgroups	20.04	40.14
IMDb	70.86	70.30

We then took the best of those models and tuned the ccp_alpha hyperparameter for minimal cost-complexity pruning by comparing training and testing accuracies, as shown in Figures [6] and [7].

We noticed that without using minimum cost complexity pruning, if we set the max_depth hyperparameter to the same depth as the optimally pruned trees, the models are less generalizable, shown by the lower test accuracies:

Parameter Name	**ccp_alpha**		**max_depth**	
Accuracy Type	**Training**	**Testing**	**Training**	**Testing**
20 Newsgroups	57.00	43.15	83.81	39.58
IMDb	78.63	74.66	93.83	71.48

For 20 Newsgroups, the optimal ccp_alpha was determined to be $3.4659 * 10^{-4}$ based on Figure [6]. This tree had a depth of 144 nodes. For IMDb, the optimal ccp_alpha was found to be $3.5380 * 10^{-4}$ based on Figure [7], yielding a tree with 29 nodes.

Random Forest

Our optimal models were found by training many separate models, 30 of which are shown in Figures [10] and [11].

These figures also show the difference in OOB validation accuracy when we use different numbers of input parameters (square root, logarithm, or all) when training each individual tree. We see that the error rate is higher when more input parameters are used. We also see that the model trained using entropy as the splitting criterion did not perform as well as the equivalent model trained using the Gini index. The most important parameter in these figures is the number of trees trained in each forest: the figure shows the models with 1 tree, 75 trees, and 150 trees, with the error rate decreasing with the number of trees. We also noticed that training each tree on all features caused the runtime of fitting the model to be exponentially longer.

Figures [12] and [13] show the validation error and test error over the number of trees in the forest. From these plots, we determined that we should use n_estimators = 500 for 20 Newsgroups and n_estimators = 650 for IMDb, as these mark where the validation plots flatten.

We also looked at forests trained without bootstrapping on the 20 Newsgroups dataset (i.e. every tree sees the same data points). The non-bootstrapped 5-fold CV training error (36.88%) was much higher than the bootstrapped CV error (2.56%), likely owing to the increased correlation between the weak learners in the non-bootstrapped model.

BERT

The best model was found by running 3 epochs on a learning rate of 2×10^{-5} for each datasets. The algorithm was too long to run for 5-fold cross-validation so we only used 1-fold cross-validation. We used the parameters recommended by the authors of BERT (Devlin et. al, 2018). Lastly, we used the one cycle policy (Smith, 2018) to find good learning rates.

Discussions and Conclusions

We investigated the weight of the words after transformation by tfidf via plots (figure [1]).

The advantage of using *logistic regression* is its highly interpretable nature. In figure [2], we plot twenty of the highest and lowest coefficients of the model fitted on IMDb. We can interpret the words corresponding to high coefficients as positively correlated to a positive review, and the opposite for negative coefficients. In particular the presence of a word will increase the log-odds by exactly its corresponding coefficient. We observe that the mean standard deviation CV score and testing accuracy are relatively similar indicating little to no overfitting. Lastly, we observed that L2

regularization yields better test accuracies than L1. We can conclude if the goal is to extract discriminative words of each class, one should choose L1 (to induce sparsity), and if the goal is to obtain the best predictive performance, the non sparsity-inducing L2 penalty is preferred.

We observe that SVM is robust towards changing the slack hyperparameter C for both datasets (see figure [3]). In theory, increasing C results in overfitting. This indicates little regularization is needed to find the best model. The standard deviation is low and the mean CV score of the best model is relatively similar to test accuracy which implies the chances of overfitting is very small. To see if we could improve the SVM model, we increased complexity by trying the RBF Kernelized SVM. By increasing complexity, we lose the interpretability of the model, and in theory, gain predictive power. In addition, we notice that the model becomes more sensitive to the slack parameter if the RBF kernel is used (see figure [4]). As a result, tuning the parameters of the kernel SVM is more complex and empirical due to the sensitivity and the uninterpretable nature of the model. We observe similar accuracies compared with the unkernelized SVM. This points towards trying a different kernel, or trying a complete different model (i.e Adaboost, etc.).

Adaboost with decision stumps requires two hyperparmeters to tune - the number of stumps used and the learning rate. The improvement of the model plateaues at 150 stumps regardless of learning rate (see figure [5]). The standard deviation of the CV scores are very low. In both datasets, the mean CV score of the best estimator is very similar to the test score which supports the fact that overfitting is not an issue even though boosting methods may increase chances of overfitting.

In the decision tree models, we noticed that the tree depth of the 20 Newsgroups model is much shallower than the IMDb model, likely because IMDb is a binary classification problem whereas the 20 classes for 20 Newsgroups required higher complexity to capture more decision boundaries. It is also evident in Figure [9] that pruning the tree decreases the depth and the number of nodes, gradually increasing the impurity of the leaves as shown in Figure [8]. This is reflected as an increase in model generalizability, as seen by the increase in test accuracies in Figures [6] and [7] despite the decrease in training accuracy. However, over-pruning the tree causes the test accuracy to decrease because the bias becomes too large.

For the random forests, we chose the number of trees for the best models by looking at where the validation error curves flatten in Figures [12] and [13]. We noticed that the test error plateaued at a lower number of estimators than did the validation error. Finally, we attempted to tune other hyperparameters including ccp_alpha, the maximum tree depths, and minimum_impurity_decrease over a log-spaced range of parameters for the trees in the forest. However, these did not improve the testing accuracy and were omitted from the final model.

For BERT, we observe that it overfits the 20Newsgroups dataset. If we had more time we could attempt regularization techniques for neural networks such as early stopping, data augmentation, etc. However it performs extraordinarily well on the IMDb dataset. We reported the loss of the neural network as a function of the learning rate (figure [14]). We had tested more epochs but it did not seem to yield any better results.

In summary, BERT performed the best out of all the models. However, it was by far the longest to train. The next best three models are the two SVM models and logistic regression. All models performed significantly better on IMDb compared to 20Newsgroups. This indicates that 20Newsgroups data is likely not sufficient to train models and one possible solution is use feature engineering methods.

Further Work

The bulk of the time spent on this assignment was manually adjusting parameters. One could try implementing automatic tuning algorithms from libraries such as Hyperopt. In addition, we only used one metric to evaluate our models on, namely accuracy. One could also evaluate the models based off the metrics - precision, recall, sensitivity, and even compute the confusion matrices for each model.

Statement of Contributions

Andrew Cheng worked on data preprocessing, BERT, logistic regression, linear and kernelized RBF SVM, and Adaboost. Chris Rafiaa worked on BERT and data preprocessing, as well as attempted to run our code on google cloud compute engine. Stephanie Li completed random forest and decision trees. All three members contributed in writing the report.

Citations

1. Devlin, J., Chang, M.-W., Lee, K., Google, K., Language, A. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/pdf/1810.04805.pdf

2. Lecture 14: Kernels continued. (2020). Cornell.Edu. https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote14.html

3. Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. ArXiv.Org. https://arxiv.org/abs/1803.09820

1 Appendix

```
Features with lowest tfidf:
['suplexes' 'intercontinental championship' 'investigation to'
 'came to his' 'do have problem' 'vampire cloak' 'to dying'
 'forget about any' 'his initial' 'non other' 'nearly kills' 'is impaled'
 'ring for' 'him and to' 'on that later' 'consider for' 'about any gore'
 'produced directed by' 'with supposed budget' 'supposed budget']
Features with highest tfidf:
['hackenstein' 'tanner' 'lennon' 'coop' 'xica' 'shearer' 'zatoichi'
 'zizek' 'gadget' 'darkman' 'demons' 'khouri' 'botched' 'woo' 'more more'
 'casper' 'wei' 'dev' 'scanners' 'pokemon']
```

Figure [1]: The twenty highest and lowest tfidf weighted features in the IMDb dataset.

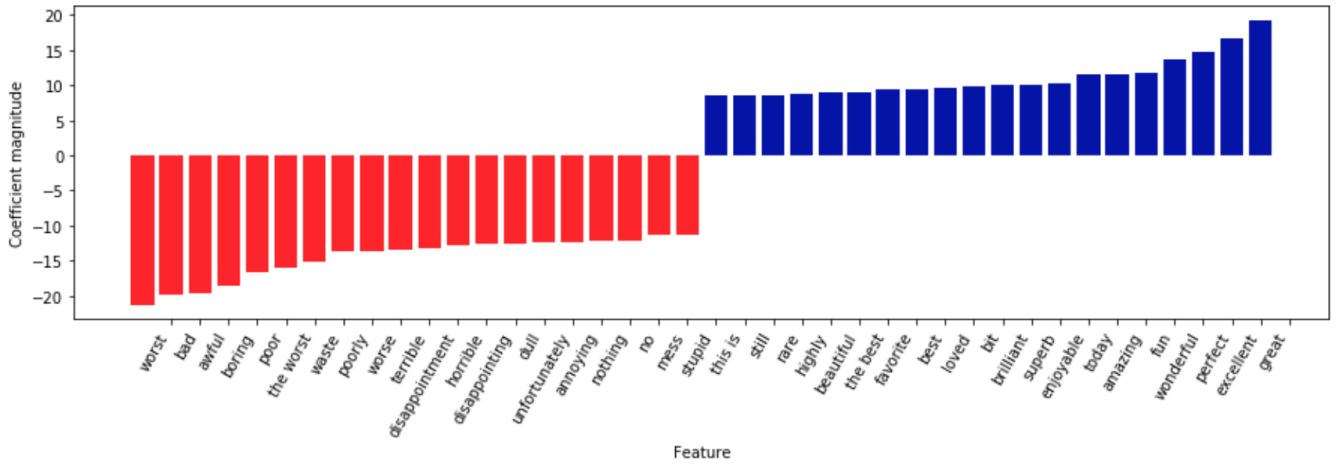


Figure [2]: The highest and lowest coefficients of logistic regression fitted on IMDb and their respective features.

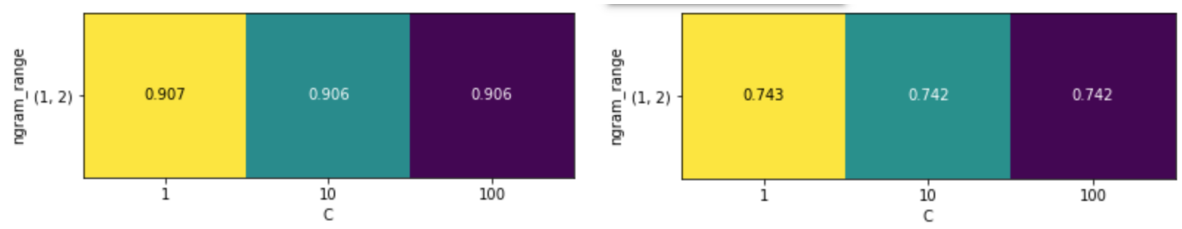


Figure [3]: SVM is robust in response to changing slack hyperparameter C. The results shown are the mean of the 5-fold cross validation scores of each fitted model.

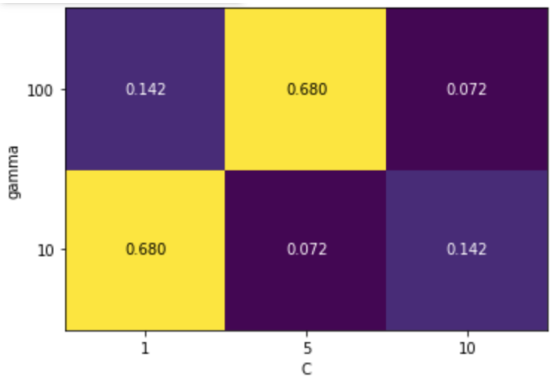


Figure [4]: RBF kernelized SVM is highly sensitive to different parameter of the slack variables per fixed gamma value which leads to difficult hyperparameter tuning (20 Newsgroups dataset).

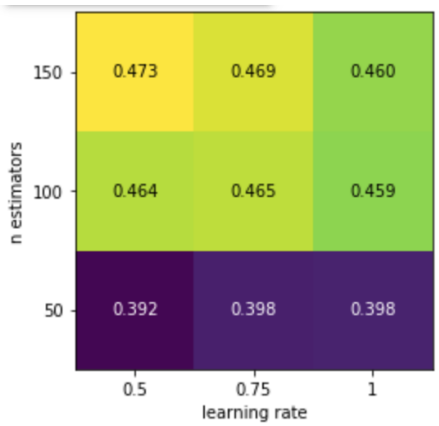


Figure [5]: The performance of Adaboost (tree stumps) plateaus at 150 estimators.

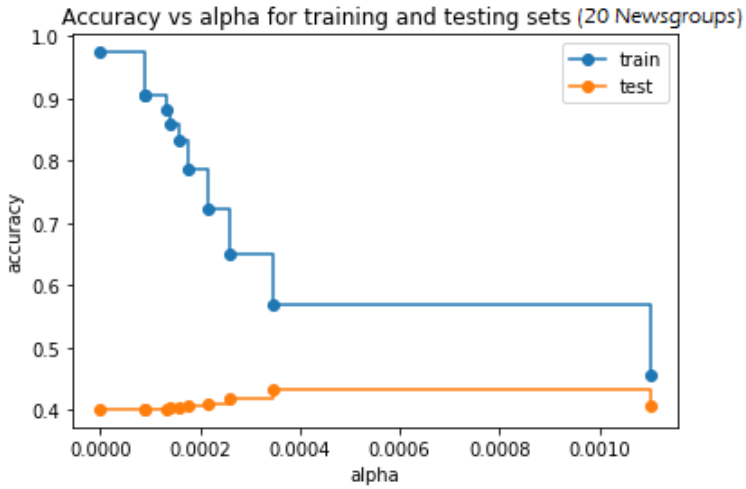


Figure [6]: The accuracy of the decision tree model over ccp_alpha values on the 20 Newsgroups dataset.

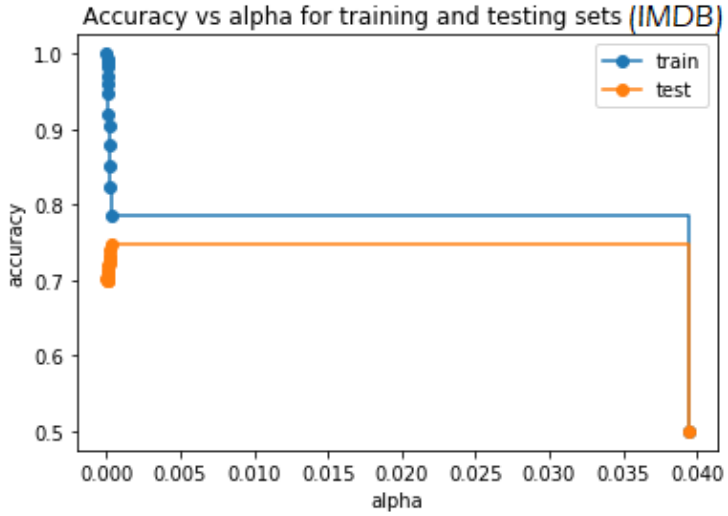


Figure [7]: The accuracy of the decision tree model over different ccp_alpha values on the IMDb dataset.

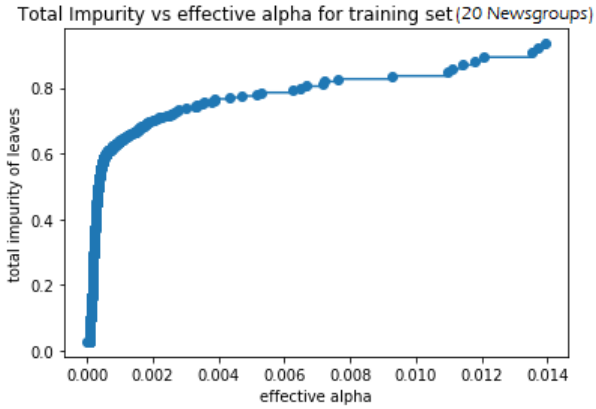


Figure [8]: The impurity of the remaining leaves of the decision tree over ccp_alpha values on the 20 Newsgroups dataset. A similar plot was obtained for IMDb (not shown).

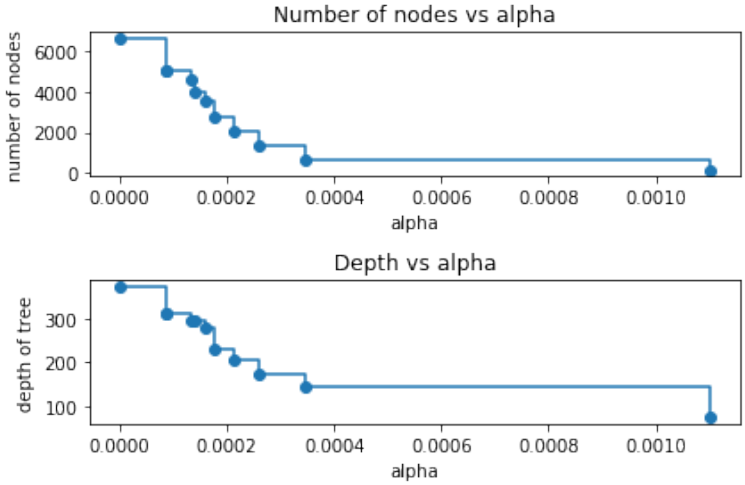


Figure [9]: The number of nodes and depth of the decision tree plotted against the ccp_alpha value, showing the pruning process. A similar plot was obtained for IMDb (not shown).

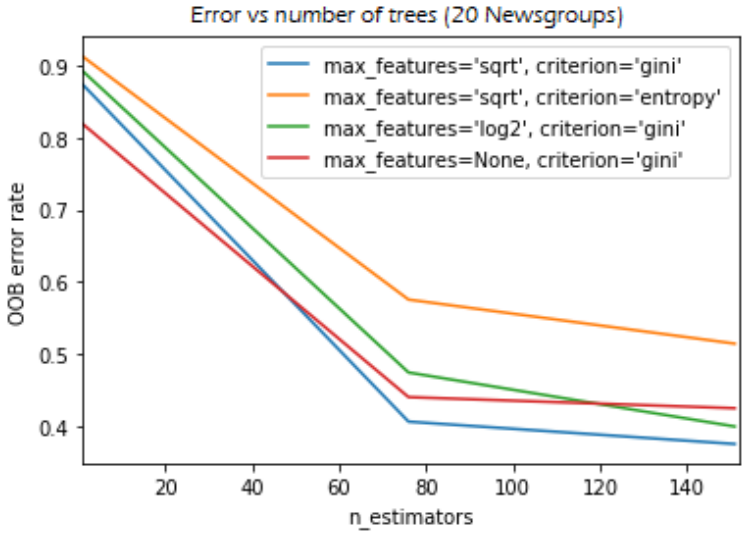


Figure [10]: The OOB error over different selected decision tree models that we tested for 20 Newsgroups. The max_features hyperparameter toggles the number of sampled features for each tree.

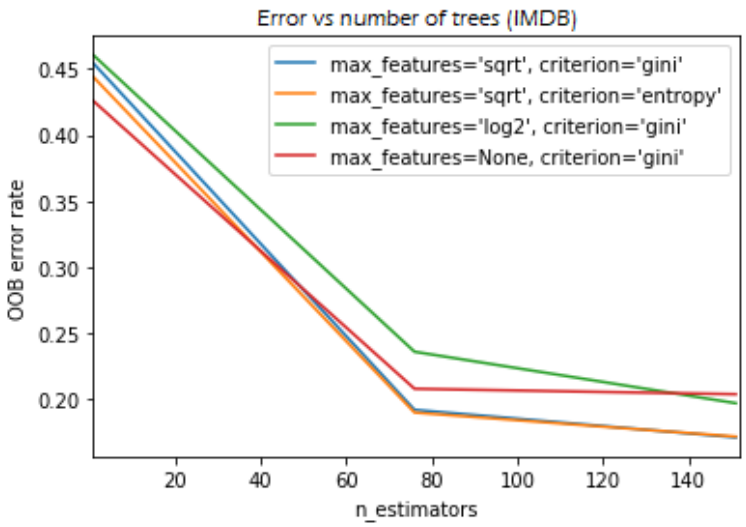


Figure [11]: The OOB error over different selected decision tree models that we tested for IMDb. The max_features hyperparameter toggles the number of sampled features for each tree.

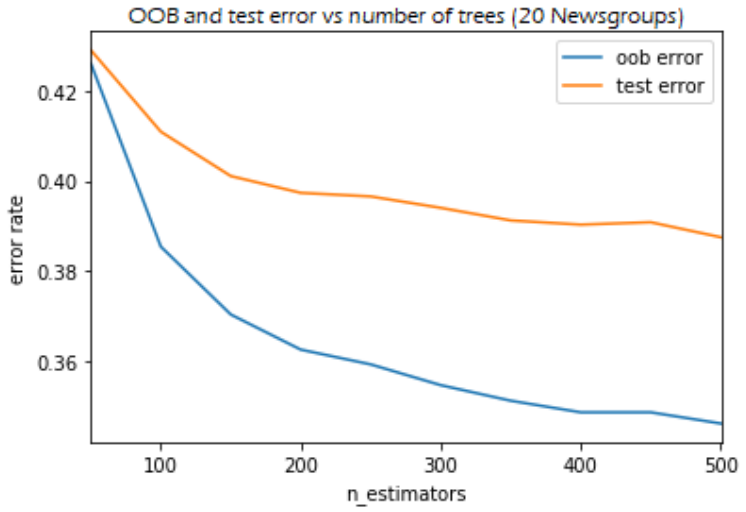


Figure [12]: The OOB and test error for our best random forest model (criterion='gini', max_features='sqrt') over the number of trees in the forest for 20 Newsgroups. The errors were also plotted for forests with more than 500 trees, but improvements were not found in the validation error so the data was not shown.

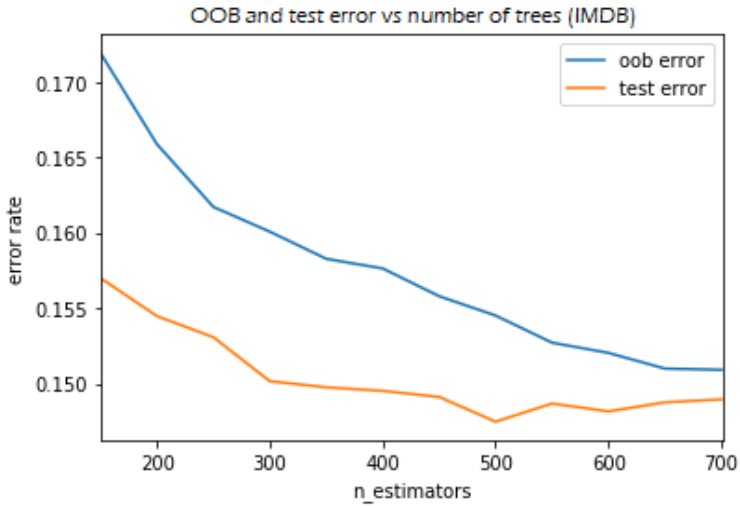
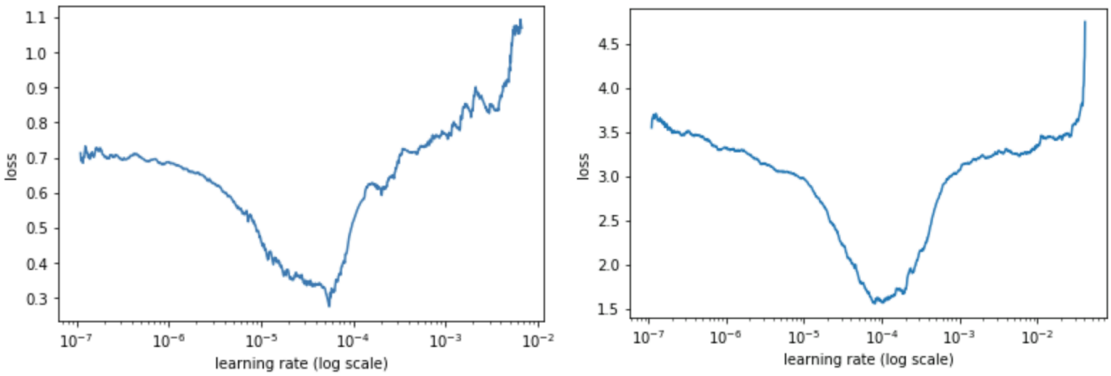


Figure [13]: The OOB and test error for our best random forest model (criterion='gini', max_features='sqrt') over the number of trees in the forest for IMDb.



Figure[14]: Loss as a function of learning rate for BERT. The plot on the left is IMDb and right is 20Newsgroups.