Analysis of Supervised Classification Models on Multi-class Textual Data

Hong Kun Tian, Eric Shen and Suzy Liu

Abstract— **The aim of this project is to train classification models and then evaluate and assess their performance on classification of textual data. We trained five models: Logistic Regression (LR), Decision Tree (DT), Support Vector Machines (SVM), AdaBoost, and Random Forest (RF) on two datasets: 20 newsgroups and IMDB reviews, from scikit-learn and Stanford University[1] respectively. The accuracy of the models is used as the metric to evaluate the models' performance. We also vectorize all datasets, use tf-idf term weighing to process our data and improve performance. As a result we find SVM gives us the overall best accuracies on both datasets among all the models we trained.**

I. Introduction

Preliminaries:

Logistic Regression, Decision Tree, Support Vector Machine, Ada Boost and Random Forest are a few of the many classification models used in Machine Learning[2]. We implemented these five models using the scikit-learn package in order to classify textual data for two datasets. For the given two datasets, 20 newsgroups and IMDB reviews, the input are text documents. By applying the five models above, we are able to output the data as categorical variables.

Project goal

In this project, we planned to compare the performance of five models mentioned in preliminaries and get the accuracy of the best model we developed and derive the combined accuracy of two datasets. In order to reach our goals, we first pre-processed the aforementioned datasets. We downloaded the datasets and checked their class distribution. We then vectorized each dataset using scikit-learn's CountVectorizer() followed with scikit-learn's TfidfTransformer() (For clearer code, we opted to use TfidfVectorizer() later on since it performs the same process but in a single step). In order to maximize performance, we performed some feature selection through the exploration of using some different parameters for TfidfVectorizer(). This is discussed in detail in the Data Set and Setup section. We then built the models and validation pipelines. We performed hyperparameter tuning on all the models in order to maximize performance. We trained a default model and a tuned model for all five models on the given two datasets. We then compared the default model and the tuned model on the test datasets for both datasets and derived the test accuracy, which helped us the plot the confusion matrix graph for the results.

Important findings

After applying the models on the given two datasets, we compared the accuracy result derived from those models and found from our analysis that SVM is an extremely scalable and fast model. In our experiments, SVM accomplishes a top accuracy of 0.85316 on unseen test data. On the other end of the spectrum, based on our results, Decision Tree and AdaBoost seem to perform poorly for multi-class classification. The two models had top accuracies on unseen test data of 0.56014 and 0.54806 respectively.

II. Related work

While we were looking for the performance of all five models on other works, we found an article about multi-class text classification using scikit-learn[3]. In this article, Susan Li applied Logistic Regression, SVM and Random Forest on a Consumer Complaints dataset, where all data entries in the dataset are text-based, and she compared the performance of these models. The result for SVM is 0.822890 while Logistic Regression and Random Forest has result of 0.792927 and 0.443826 respectively. Though this paper only includes three of the models we have, but we still noticed a strong similarity between our results and hers. Notably, SVM has the best performance among all the models.

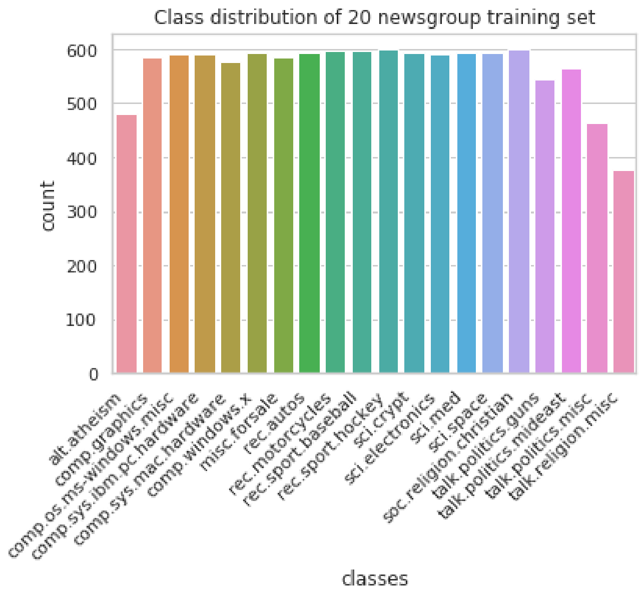

Fig. 1: 20 news training set class distribution

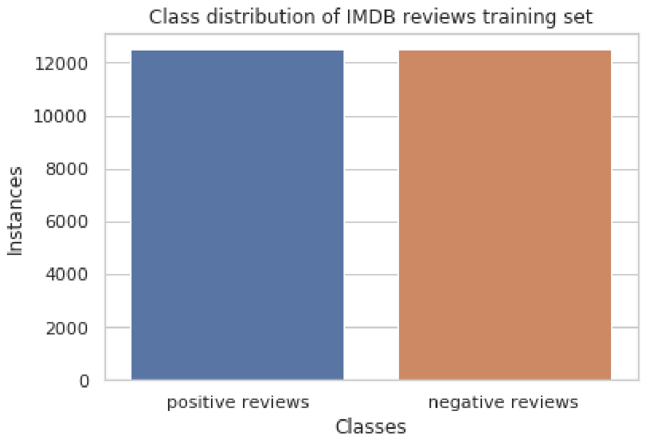

Fig. 2: IMDB reviews training set class distribution

III. Dataset and setup

As fig 1, shows, 20 news group training set has 11314 instances, each of them contains 20 classes.

As fig 2, shows, IMDB review traning set [1] has 25000 instances in train set and 25000 instances in test sets.We only analysis train set, and there is 12500 positive results and 12500 negative results. Our goal is to predict the results are either positive or negative in test set.

For both datasets, we extract the features from the textual data using scikit-learn's CountVectorizer() on the training set with the default parameters. scikit-learn's CountVectorizer class performs word tokenization and vectorizes the features. We depended on scikit-learn to one-hot encode both datasets. Furthermore, we perform tf-idf term weighing on the extracted textual features from CountVectorizer using TfidfTransform(). Tf-idf is short for term frequency–inverse document frequency. This step ensures that extremely frequent words that don't necessarily convey meaningful information don't overshadow more interesting and predictive terms that occur less frequently. More specifically, the tf-idf is calculated as follows:

$$\text{tf-idf} = \text{times term } t \text{ appears in doc } d * \text{idf}(t)$$

where $\text{idf}(t)$ is calculated using:

$$log(\frac{\text{total no. of docs}}{1 + \text{total no. of docs } d \text{ containing term } t})$$

As a side note, there exists the class TfidfVectorizer that performs both these steps consecutively. During feature selection which is detailed below, we used the latter.

IV. Proposed Approach

Feature Choice

For the feature selection process, we decided to use scikit-learn's TfidfVectorizer() and its min_df parameter. The min_df parameter specifies during the CountVectorizer step which terms to ignore given that they have a document frequency lower than the given threshold. If given a float, the parameter represents a proportion of the documents. If given an int, it represents an absolute count.

We found this particular parameter extremely useful because this allows us to remove a large number of non-meaningful features to improve training time for some learning models. Additionally, it helps prevent our models from overfitting since we will be removing terms that occur too rarely to be useful when using unseen data. We test different values of min_df for each model dataset pair and run a 5-fold cross validation for each value to ensure that we choose the min_df that yields the highest validation score. In some cases, despite not performing better than the default TfidfVectorizer, we opted to still choose the highest performing min_df because of the potential decreased training time it may yield due to removing a large proportion of the features. This

is done in consideration for the hyperparameter tuning we planned on performing. There exists also a max_df parameter that does the inverse. However, we chose to not make use of it because tf-idf weighing already takes care of high frequency terms.
Some other techniques we considered but decided against using include lemmatization, because we feel that combining words with similar roots may end up obfuscating the sentiment for certain terms. This is especially true for words that have similar roots but completely opposite meaning.

Model Choice

Logistic Regression: LR is a discriminative learning method that is trained with the data to learn the conditional distribution

$$P(Y \mid X)$$

directly. To nd such a series of weights, so that linear approximation in

$$(W^T X)$$

maximize the most likelihood function, initial weights are adjusted through the gradient descent process[4].

Support Vector Machine: SVM is a supervised learning model which aims to find a best hyperplane with (n-1) dimension to separate different samples(make a classification). In our experiment, Linear SVM is adopted to investigate the problem[5].

Decision Tree: Decision Tree is a non-parametric supervised model in machine learning, and it is used for classification and regression. Decision tree learns from the data entries in order to predict the value of a target variable[6].

Ada Boost: Ada Boost is a machine learning meta-algorithm, which can help data scientists to combine several weak classifiers into a strong classifier[7].

Random Forest: Random forest is a meta estimator in machine learning, and it can be used for classification and regression. Random forest can fit decision tree classifiers on various sub-samples of the dataset, which will later derive the average to improve the accuracy and control over-fitting[8].

Hyperparameter tuning

For our hyperparameter tuning process, we opted to use scikit-learn's GridSearchCV and RandomizedSearchCV. We performed a few searches with GridSearchCV and quickly realized that it would not been feasible to continue to use GridSearchCV because not only is it extremely expensive to train, but the results heavily rely on our ability to know a good range within which the best parameter may lie. Upon further investigation, we found James Bergstra and Yoshua Bengio's paper[9] that "shows empirically and theoretically that randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid". Therefore, we decided to abandon GridSearchCV and opted to only use RandomizedSearchCV.

V. Results

For all validation accuracies on the training sets, we use 5-fold cross validation to improve the robustness of our results. We also implemented three models to compare and find the best accuracy we can achieve.

We start by comparing all accuracies of the different default models without feature selection as shown in TABLE I. For 20 newsgroups, the best validation accuracy achieved is 92.5% from SVM, and the worst, 54% from AdaBoost. For test accuracies, we got results following a similar pattern: SVM is the best at 85.3% and AdaBoost is the worst at 50.9%. For IMDB reviews, the best validation accuracy achieved is 89.5% from SVM, and the worst, 69.9% from Decision Tree. For test accuracies, LR is the best at 88.3% and Decision Tree is the worst at 69.9%.

We also compare all accuracies of different default models with feature selection as shown in TABLE II. For 20 newsgroups, the best validation accuracy achieved is 89.4% from Logistic Regression, and AdaBoost gives us the worst, at only 54.6%. For test accuracies, we got results following a similar pattern: Logistic Regression is the best at 82.6% and AdaBoost is the worst at 51.7%. For IMDB reviews, the best validation accuracy achieved is 89.5% from SVM and Decision Trees gives us the worst, at only

Accuracy for default models with no feature selection

	20 Newsgroups(Averaged validation / Test)	IMDB Reviews(Averaged validation / Test)
Logistic Regression	0.88996 / 0.82740	0.88828 / 0.88272
Decision Trees	0.63240 / 0.55217	0.70312 / 0.69888
Support Vector Machines	0.92514 / 0.85316	0.89472 / 0.8776
Ada boost	0.54605 / 0.50876	0.80324 / 0.80296
Random Forest	0.83507 / 0.76129	0.83548 / 0.83752

TABLE I: We use 5-fold validation with all averaged validation

Accuracy for default models with feature selection

	20 Newsgroups(Averaged validation / Test)	IMDB Reviews(Averaged validation / Test)
Logistic Regression	0.89385 / 0.82581	0.88832 / 0.88288
Decision Trees	0.62516 / 0.55961	0.7012 / 0.70504
Support Vector Machines	N/A / N/A	0.89472 / 0.87692
Ada boost	0.54685 / 0.51686	0.80584 / 0.8032
Random Forest	0.83313 / 0.75358	0.83752 / 0.84264

TABLE II: We use 5-fold validation with all averaged validation. For SVM on 20 newsgroups dataset, feature selection was not performed.

Accuracy for tuned models with feature selection

	20 Newsgroups(Averaged validation/ Test)	IMDB Reviews(Averaged validation/ Test)
Logistic Regression	0.91701 / 0.84254	0.89500 / 0.88348
Decision Trees	0.63302 / 0.56014	0.73192 / 0.73604
Support Vector Machines	0.92558 / 0.85276	0.89496 / 0.88464
Ada boost	0.58547 / 0.54806	0.84092 / 0.84248
Random Forest	0.84780 / 0.77589	0.84804 / 0.85136

TABLE III: We use 5-fold validation with all averaged validation.

70.1%. For test accuracies, LR is the best at 88.3% and Decision Tree is the worst, at 70.5%.

Finally, we compare all accuracies of different tuned models with feature selection as shown in TABLE III. For 20 news group, the best validation accuracy achieved is 92.6% from SVM and AdaBoost gives us the worst, at only 58.5%. For test accuracies, we got results following a similar pattern: SVM is the best at 85.3% and AdaBoost is the worst at 54.8%. For IMDB reviews, the best validation accuracy achieved is 89.5% training set from SVM, and Decision Tree gives us the worst, at only 73.2%. For test accuracies, SVM is the best at 88.5% and Decision Tree is the worst at 73.6%.

Overall, the best test accuracy we achieved for the 20 newsgroups dataset among the 5 models is 85.3% using SVM, and the best test accuracy we achieved for the IMDB reviews dataset among the 5 models is 88.5% also using SVM. Therefore, we believe SVM is the best model among five. The combined accuracy of combined prediction and combined target is 87.73515%

VI. Discussion and Conclusion

In this project, after running all the experiments and comparing the results we got, we found that SVM is the best model to deal with both the 20 newsgroups dataset and the IMDB reviews dataset, which are multi-class textual datasets.

SVM gave us the best prediction accuracy and it was significantly faster at training as well compared to the other four models. However, there are still many possibilities for us to generate a better prediction accuracy with better choices of parameters for the models. In addition, we can adopt more complex algorithms to preprocess our data and use deep learning.

VII. Statement of Contributions

- Hong Kun: Preprocessing, LR, DT, SVM, ADA, and writeup
- Eric: Preprocessing, ADA, and writeup
- Suzy: Random Forest, ADA, and writeup

References

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "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), pp. 142–150, Association for Computational Linguistics, June 2011.

[2] J. O. E. N. E. S. Yang Cao, Xin Fang, "A comparative study of machine learning algorithms in predicting severe complications after bariatric surgery," *Journal of Clinical Medicine*, 2019.

[3] S. Li, "Multi-class text classification with scikit-learn," *Journal of Towards Data Science*, 2018.

[4] S. W. Menard, "Applied logistic regression (2nd ed.)," *ISBN 978-0-7619-2208-7*, 2002.

[5] V. V. N. Cortes, Corinna, "Support vector networks," in *Machine Learning*, pp. 273–297, 1995.

[6] P. E. Utgoff, "Incremental induction of decision trees. machine learning," 1989.

[7] "Boosting algorithms: Adaboost, gradient boosting and xgboost," 2018.

[8] T. K. Ho, "Random decision forests," 1995.

[9] J. B. James Bergstra, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, 2012.

VIII. Appendix

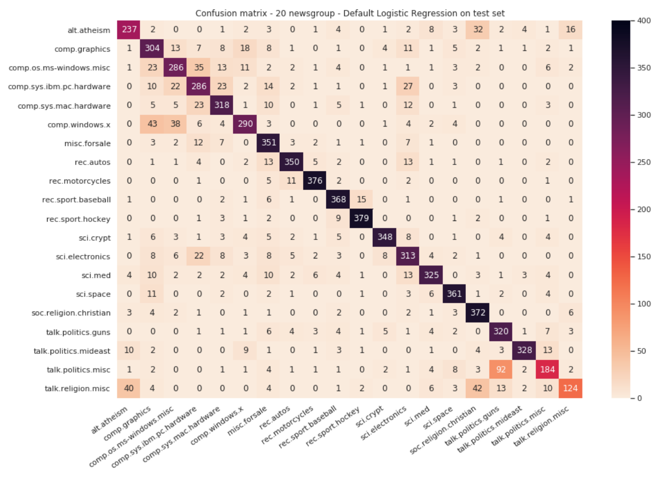

Fig. 3: 20 news testset

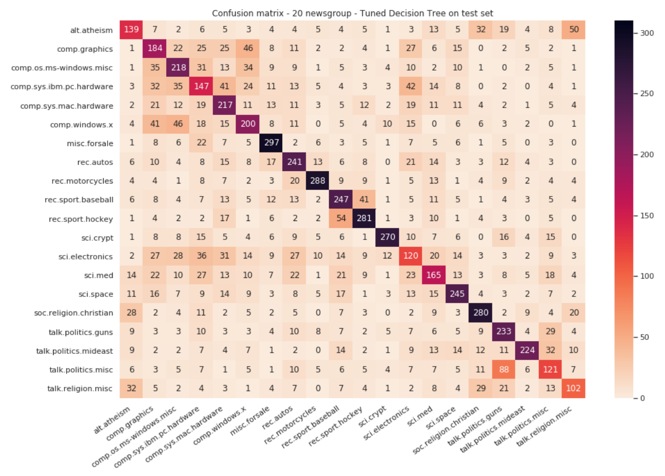

Fig. 6: 20 news testset

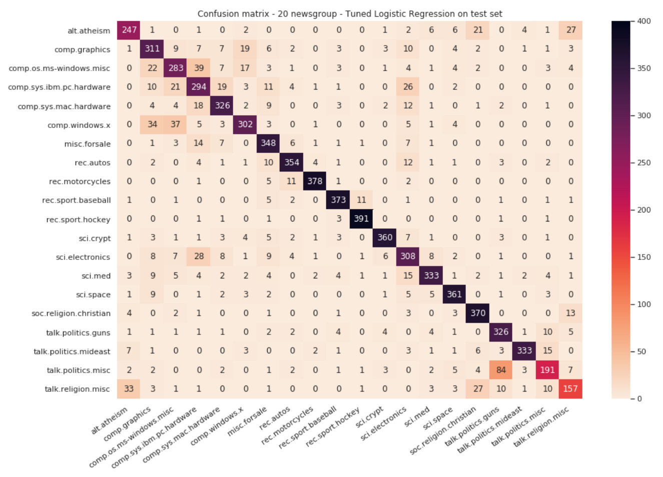

Fig. 4: 20 news testset

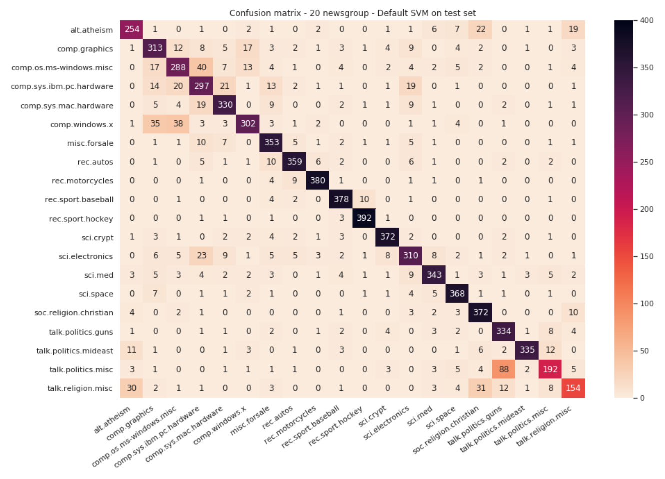

Fig. 7: 20 news testset

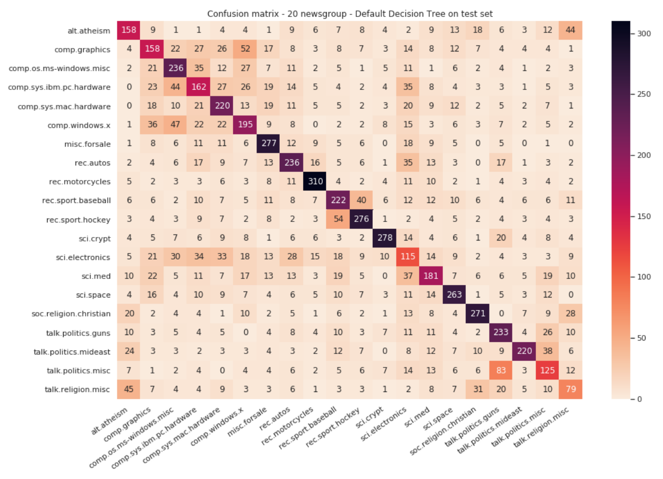

Fig. 5: 20 news testset

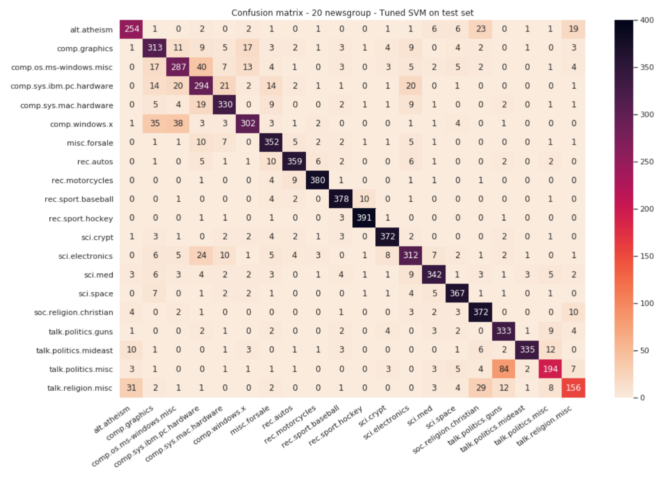

Fig. 8: 20 news testset

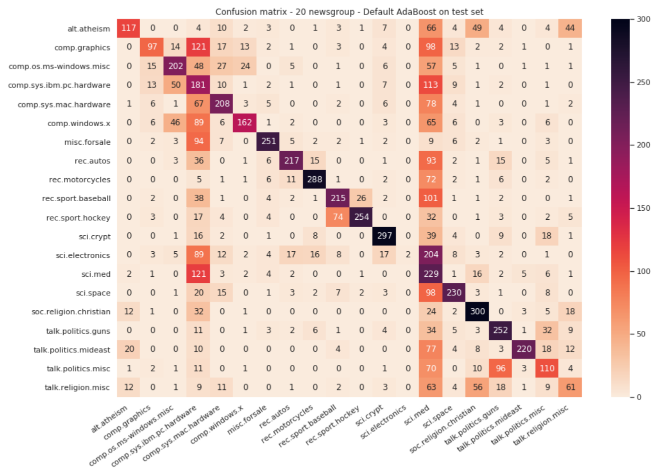

Fig. 9: 20 news testset

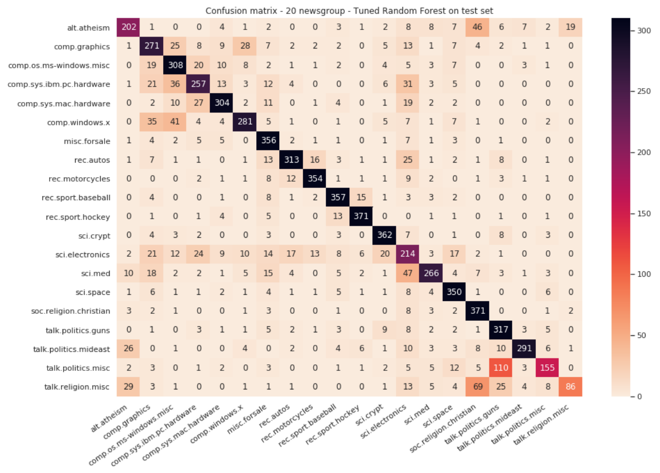

Fig. 12: 20 news testset

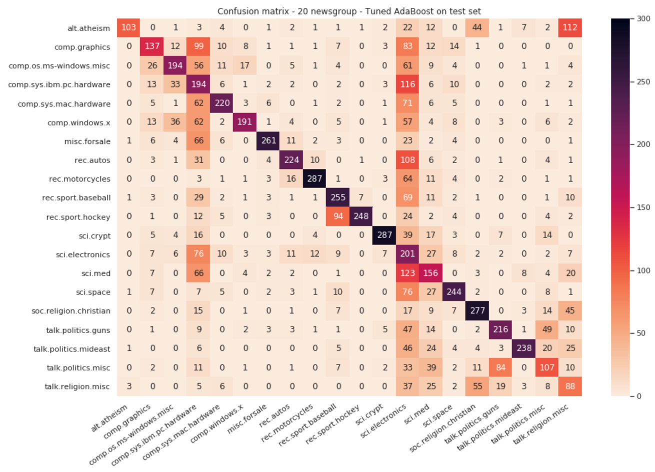

Fig. 10: 20 news testset

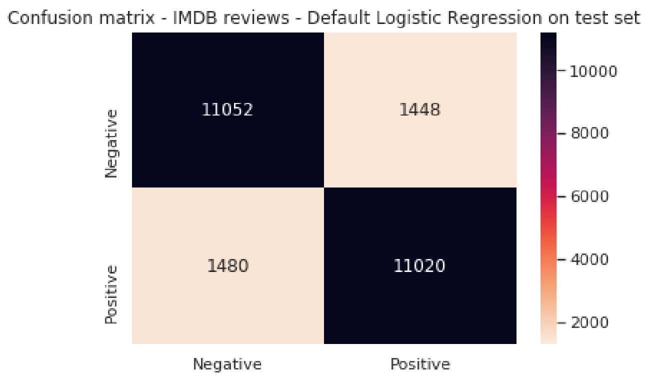

Fig. 13: IMDB testset

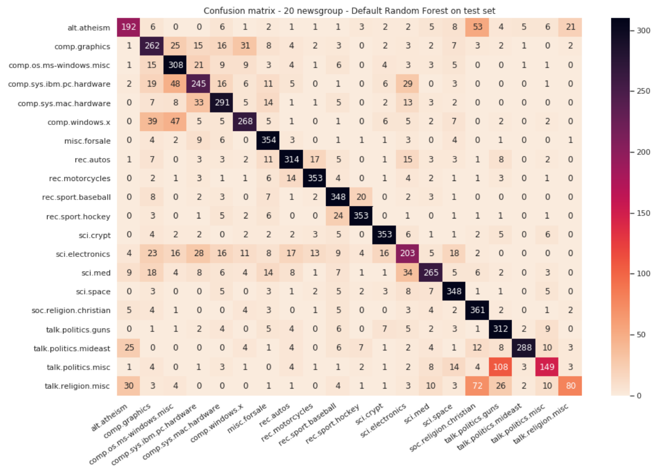

Fig. 11: 20 news testset

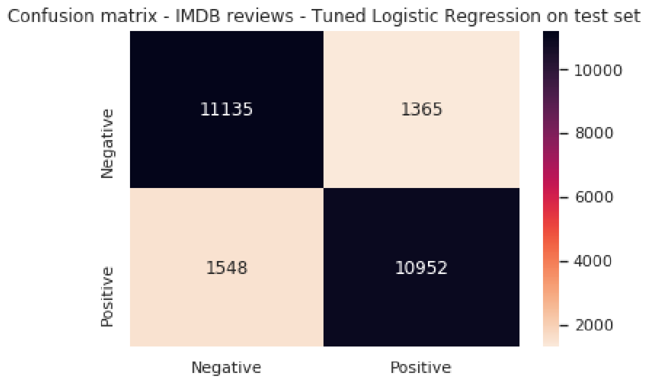

Fig. 14: IMDB testset

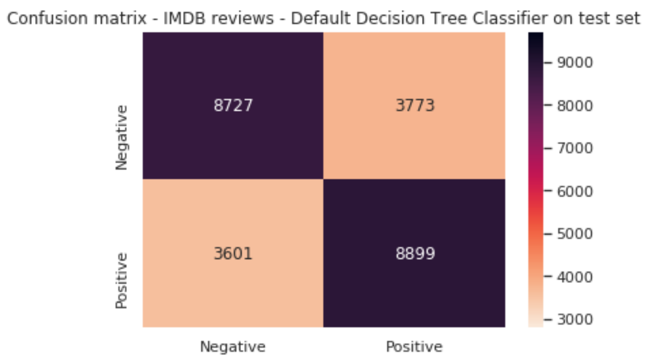

Fig. 15: IMDB testset

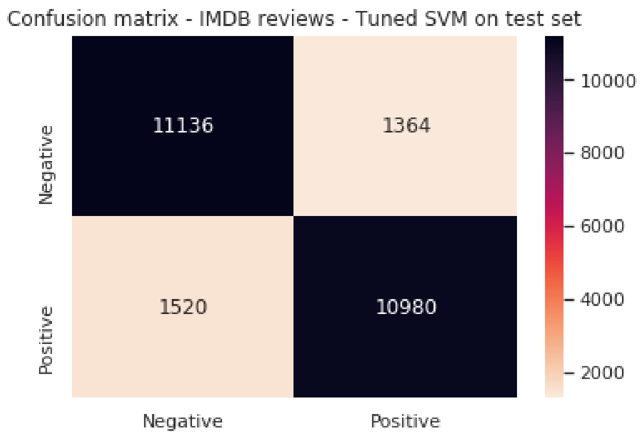

Fig. 18: IMDB testset

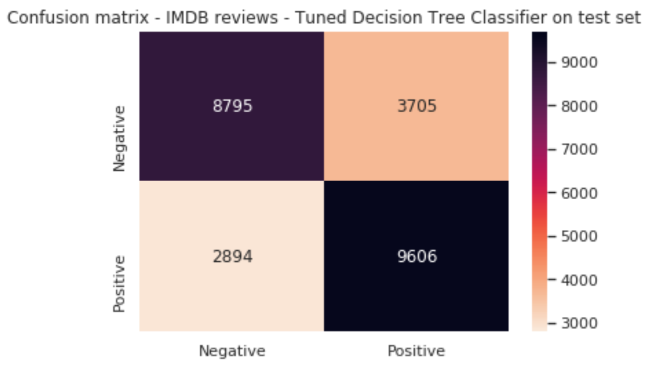

Fig. 16: IMDB testset

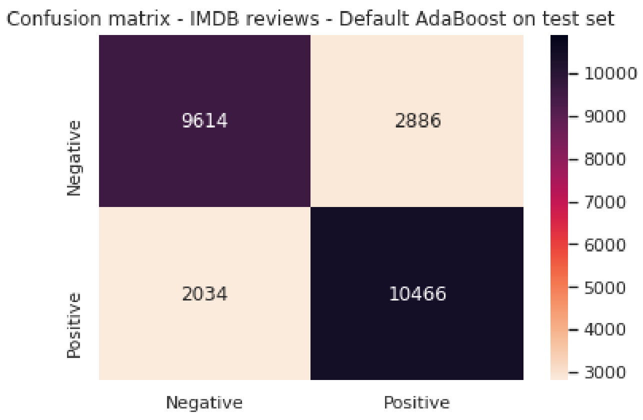

Fig. 19: IMDB testset

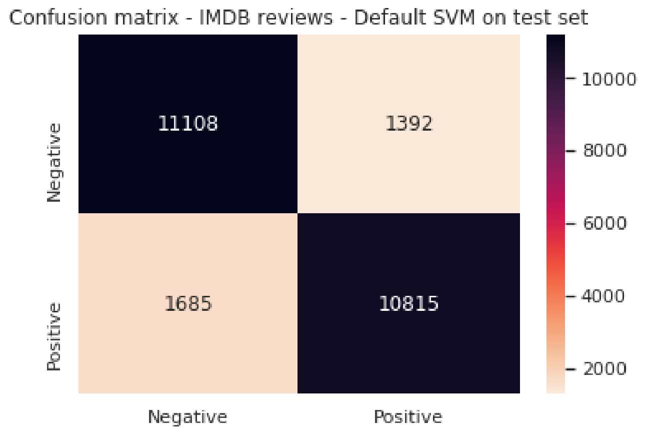

Fig. 17: IMDB testset

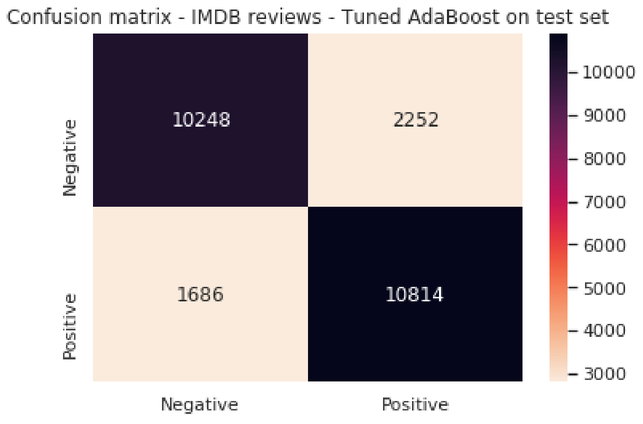

Fig. 20: IMDB testset

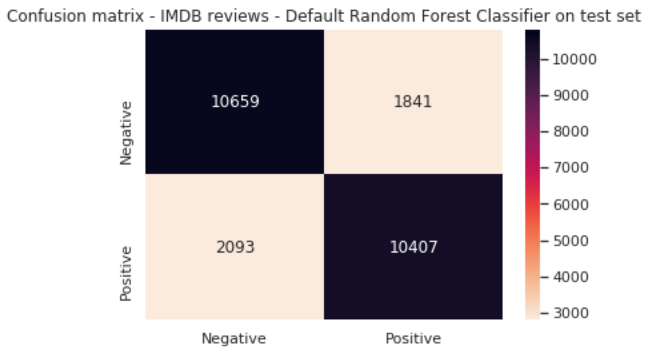

Fig. 21: IMDB testset

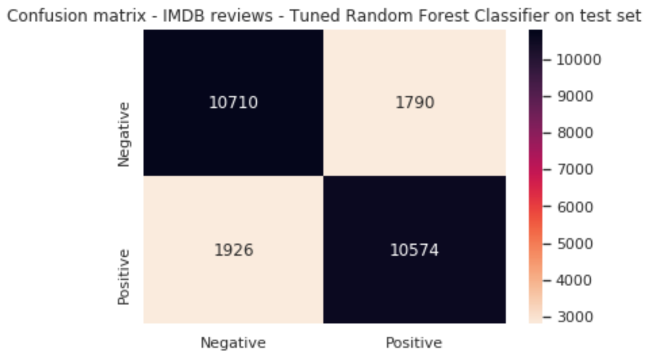

Fig. 22: IMDB testset