CIS 520 Project Report
Recommendation Prediction for Women's E-Commerce Clothing Reviews

Yixuan Meng, Zhaozheng Shen, Zhouyang Fang
School of Engineering and Applied Science, University of Pennsylvania
{yixuanm, shenzhzh, fangzhy}@seas.upenn.edu

Abstract

In recent years, review text classification has attracted the attention of many researchers, which motivate us to come up with our predictive model to classify clothing rating with supervised learning. Based on the Women's E-Commerce Clothing Reviews Dataset, we implement Support Vector Machines, Logistic Regression, Random Forest, CNN, and BERT to classify the reviews, and integrate non-text metadata into our models. The Pre-trained BERT model outperforms the baseline model with a higher accuracy of 85.91% on the test dataset.

1 Motivation

Electronic commerce (E-commerce) has attracted countless companies and customers by providing speedy access and wide variability of good. E-commerce transactions generate trillions of profits across the world, thus analysing customers' opinions is crucial to companies. Real reviews of customers are essential for investigating quality of products and customer's satisfaction, and text classification is a reliable method to tell whether customers are satisfied with the merchandise. This project aims to come up with our predictive model to classify E-commerce clothing reviews with supervised learning.

To build an effective classification model, we form this task into a two-way classification problem, and our target is to predict whether customers recommend a product based on the reviews they give. Machine learning methods can extract important features from a large amount of reviews by considering contexts while embedding and encode information of text while learning. We utilize compare the performance of our models. Notably, we also integrate non-text metadata in our models.

2 Related Work

Text classification is a machine learning technique that assigns a sentence or document to an appropriate category. There are a number of recently emerging real-life applications. One very simple example is to judge whether the email is spam based on keywords in the text. Another example is sentiment analysis, which is analyzing opinion polarity, such as positive and negative, by reading a sentence. Our problem is to build a text classification model to predict whether customers will recommend a product based on review text and other information.

There have been many studies on the application of machine learning in text classification. Some of the most popular text classification algorithms include support vector machines, Naive Bayes and deep learning.

Deep learning algorithms can learn high-level features from data, and get better results than traditional machine learning models text classification. TextCNN[4] uses the convolutional neural network, which is a useful deep learning algorithm for text classification problem.

One of the previous works on women's e-commerce clothing[1] implemented a bidirectional recurrent neural network (RNN) with long-short term memory unit (LSTM) for recommendation and sentiment classification in this data set. Results have shown that a recommendation is a strong indicator of a positive sentiment score, and vice-versa. On the other hand, ratings in product reviews are fuzzy indicators of sentiment scores.

The attention mechanism[8] has the ability to specialize the processing on different parts of the environments instead of passively processing all the data, which is intuitive enough. In addition, it offers a very powerful way to mitigate the problem with vanishing and exploding gradients for long

sequence in text classification problems. For example, HAN[9] can learn how much contribution of each word on the classification decision by adding two attention layers.

Transformer models compute without considering the sequential information, which is also very popular in NLP. Some works such as RoBERTa[6], ALBERT[5], and Transformer-XL[2] are used for text classification tasks.

3 Dataset

3.1 Background and Data Summary

We use the Women's E-Commerce Clothing Reviews Dataset[1] in this project. This dataset provides reviews written by customers. It contains 23486 rows, each row represents a customer's review. As shown in Table 2, there are 10 data fields in total, their meaning are as follow:

- **Clothing ID** is an identifier that refers to the reviewed cloth.
- **Age** is the age of reviewers.
- **Title** is a string, is the title of review.
- **Review Text** is the review body.
- **Rating** is integer ranges from 1 to 5, refers to the rank of the cloth by the customer.
- **Recommended IND** is 0 or 1, indicates whether the customer recommend the product.
- **Positive Feedback Count** refers to how many other customers think this review is positive.
- **Division Name** of the product.
- **Department Name** of the product.
- **Class Name** of the product.

[1]https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews

Column	Data Type	#Unique	Mean
Clothing ID	int64	1206	918.12
Age	int64	77	43.20
Title	object	13994	-
Review Text	object	22635	-
Rating	object	5	4.20
Recommended IND	int64	2	0.82
Positive Feedback Count	int64	82	2.54
Division Name	object	4	-
Department Name	object	7	-
Class Name	object	21	-

Table 1: Types and distributions of Data fields in the Women's E-Commerce Clothing Reviews Dataset

3.2 Data Visualization

To explain the dataset further, the distributions of length of the title and review body are shown in Figure 1. The word cloud for reviews that recommend the product is shown in Figure 2, and word cloud for reviews that speak against the product is shown in Figure 3. There are more positive adjectives like 'perfect', 'love' in the former, while the latter concerns more about 'size', 'fabric', and 'return'.

We also visualize the correlation between features, according to Figure 4 shows that 'age', 'positive feedback count', and 'review length' are not strongly correlated with our target 'Recommended IND'. Meanwhile, whether the customer recommend certain cloth and their 'Rating' are highly correlated, therefore we do not add the feature 'Rating' to our model in the experiments.

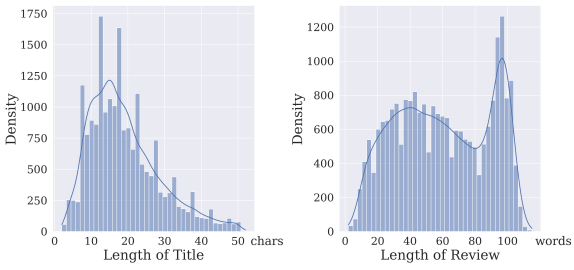


Figure 1: Distribution of Text Length

3.3 Data Pre-processing

The feature "Clothing ID" is an identifier that refers to the reviewed cloth, and it is unique for each

Figure 2: Word cloud for positive reviews.

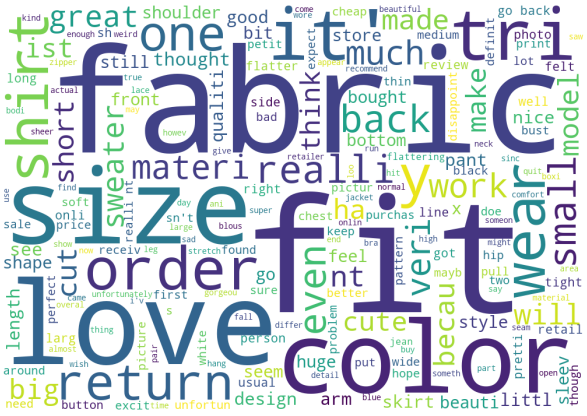


Figure 3: Word cloud for negative reviews.

product. Compared with clothing ID, we pay more attention to the number of appearances of this product. We count the number of times for each clothing ID and generate the corresponding new feature: "Clothing Count".

There are three categorical variables: "Division Name", "Department Name", and "Class Name", we created one-hot vectors for these categorical variables in the dataset.

For text features "Title" and "Review Text", considering the output of word embedding might be high-dimensional features, which is quite unfriendly to machine learning methods and might cause the problem of curse of dimensionality, we extract two features instead of word embedding directly in machine learning algorithms.

The first feature is the length of the entire Review Text, and the second feature is sentiment analysis. Firstly we remove stop words, such as "the", "a", "an", "in", to save computing time and space. Then we do sentiment analyse using *nltk.sentiment.vader.SentimentIntensityAnalyzer*.

Figure 4: Correlation matrix for selected features.

4 Problem Formulation

Our problem is a supervised binary classification problem. We predict whether the reviewer recommended the product using the rest of the data except rating, which too closely correlates with our goal, making other features almost no use. We use *Precision*, *Recall*, *F1*, *ACC*, *AUC* to measure the performance of each method from different perspectives.

We split our dataset into two big parts: training set (90%) and testing set (10%). For those traditional machine learning models, 3-fold was used to tune parameters in the training set. After picking the best parameters, model was re-trained on the whole training set and ready to compare with others on testing set. For AutoML, CNN, and Pre-trained BERT, training set was split into training part (90%) and validation set(10%). The reasons would be given in their experiments sessions.

5 Method

We explored several traditional machine learning models, as well as some powerful deep learning methods. Besides, we also tried a brand-new but promising AutoML model[3] which is still experimental.

5.1 Logistic Regression (Baseline)

sklearn.linear_model.LogisticRegression
https://scikit-learn.org/stable/modules/generated/sklearn.linearmodel.LogisticRegression.html

Logistic regression is a basic statistical model using logistic function to model a binary dependent

variable, although many more complex extensions exist. The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification, but it can be used to make a classifier by choosing a cutoff value.

5.2 Decision Tree

sklearn.tree.DecisionTreeClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

5.3 Random Forest

sklearn.ensemble.RandomForestClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

5.4 Support Vector Machine

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

SVMs are one of the most robust prediction methods based on statistical learning frameworks. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier

5.5 XGBoost

xgboost.XGBClassifier
https://xgboost.readthedocs.io/en/latest/python/python_intro.html

Salient features of XGBoost which make it different from other gradient boosting algorithms include:

- Clever penalization of trees
- A proportional shrinking of leaf nodes
- Automatic Feature selection

5.6 AdaBoost

sklearn.ensemble.AdaBoostClassifier
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

5.7 AutoML

autosklearn.experimental.askl2.AutoSklearn2Classifier
https://automl.github.io/auto-sklearn/master/api.html

As the next generation of *autosklearn.classification*, *AutoSklearn2Classifier* includes latest research on automatically configuring the AutoML system itself and contains a multitude of improvements which speed up the fitting the AutoML system.[3]

Automated machine learning (AutoML) is the process of automating the tasks of applying machine learning to real-world problems. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model.

5.8 Convolution Neural Network

A neural network imitates neurons and is widely used in classification problems. As a type of neural network, CNN takes an input vector and passes it to hidden layers that consist of three-dimensional neurons. The output of the hidden layers is passed to a fully connected layer, which generates a binary class. We use a matrix of embedding vectors to encode the review text and use convolution layers to extract the dependency of the vectors. The metadata vectors are concatenated with the output of convolution layers and fed to the fully connected layer to output a prediction.

5.9 Pre-trained BERT model

BERT(Bidirectional Encoder Representations from Transformers) is a pre-trained language model based on a multi-layer bidirectional transformer encoder. Since it is pre-trained with a large corpus, fine-tuning it for our task is efficient. We used a

large uncased BERT model from the Huggingface, and use the tokenizer to generate a masked input. Similar to what we do for CNN, we concatenate the metadata vector with the output of the BERT model and use a fully connected layer to map the vector to a binary output.

6 Experiments and Results

6.1 Helper Functions

6.1.1 Successive Halving Grid Search

sklearn.model_selection.HalvingGridSearchCV
sklearn.experimental.enable_halving_search_cv
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingGridSearchCV.html

HalvingGridSearchCV() gives far better performance than *GridSearchCV* since it search over specified parameter values with successive halving. Firstly it runs all parameters on a small training set, and pick $\frac{1}{3}$ best-performed parameters to run them on bigger training set. After several iterations, it runs just a few most-likely parameters on the whole set and pick the best. This way we can largely expand our search regions in given time.

It is still experimental for now and needs *enable_halving_search_cv* flag to enable. We use it to search best parameters on potential candidates. The search strategy starts evaluating all the candidates with a small amount of resources and iteratively selects the best candidates, using more and more resources.

6.1.2 Parallel Computing

sklearn.utils.parallel_backend
https://scikit-learn.org/stable/modules/generated/sklearn.utils.parallel_backend.html

Trying different hyperparameters for one model is highly parallelizable, so we want to run them concurrently to expand our search regions in certain time.

6.2 Traditional Machine Learning Models

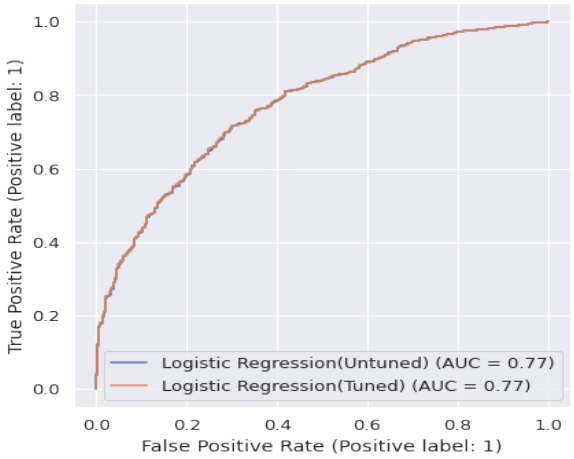
HalvingGridSearchCV() intrinsically uses *n*-fold to pick parameters. Here we use 3-fold to balance the performance and training time.

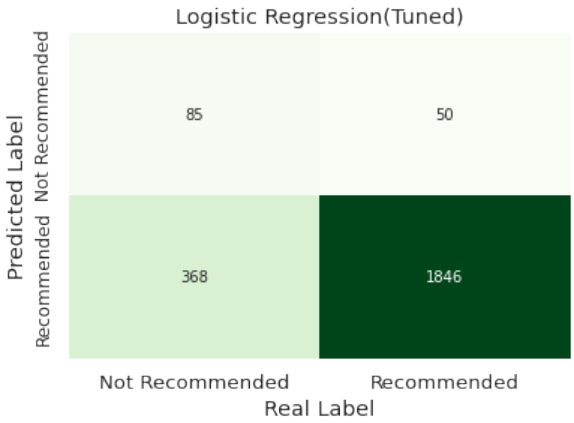
6.2.1 Logistic Regression

The hyperparameters space we search and the best results are showed below.

- penalty: **l2**, none
- C: 0.1, 0.2, 0.5, 0.7, 1, 5, **10**, 50, 100
- solver: newton-cg, **lbfgs**, sag, saga

Prediction and scores are as follow.





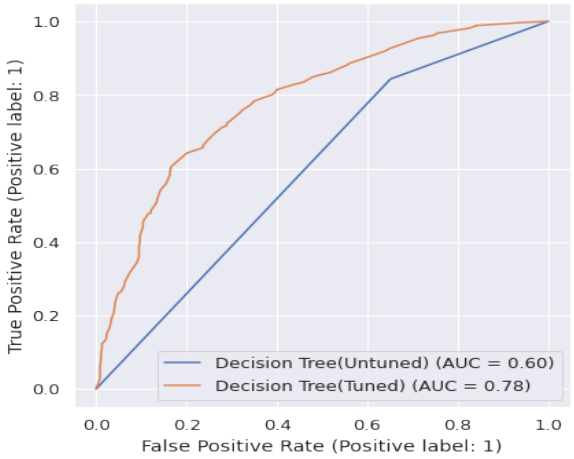
6.2.2 Decision Tree

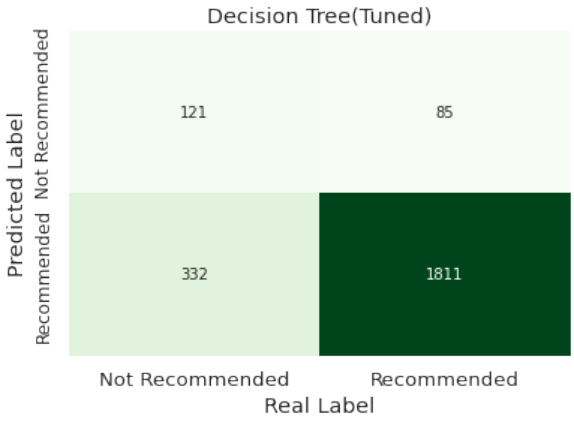
The hyperparameters space we search and the best results are showed below.

- max_features: **auto**, sqrt, log2
- splitter: best, **random**
- min_samples_split: 40, 50, 100, **200**, 500, 1000, 2000, 5000
- min_weight_fraction_leaf: 0, **0.001**, 0.01, 0.1

To be adding...

Prediction and scores are as follow.



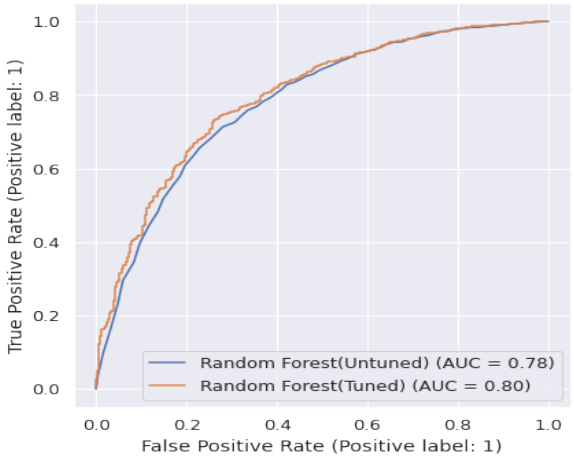


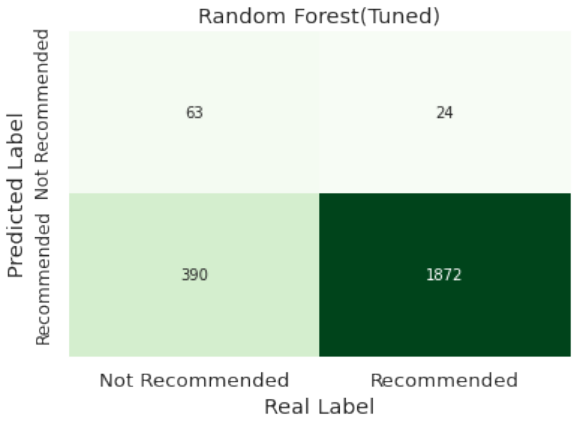
6.2.3 Random Forest

The hyperparameters space we search and the best results are showed below.

- n_estimators: 100, 200, **500**
- min_samples_split: 2, 5, **10**
- max_depth: 10, **50**, 100, 1000

Prediction and scores are as follow.



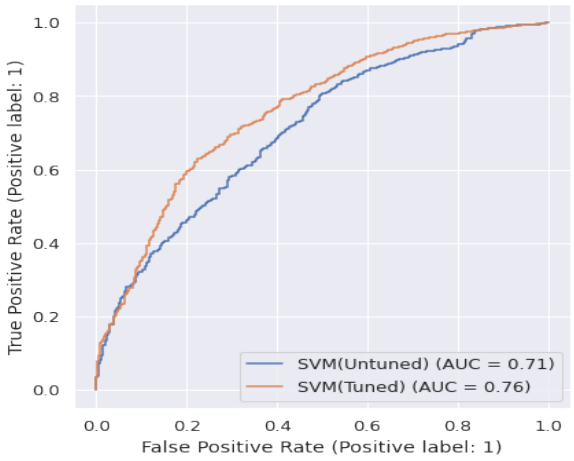


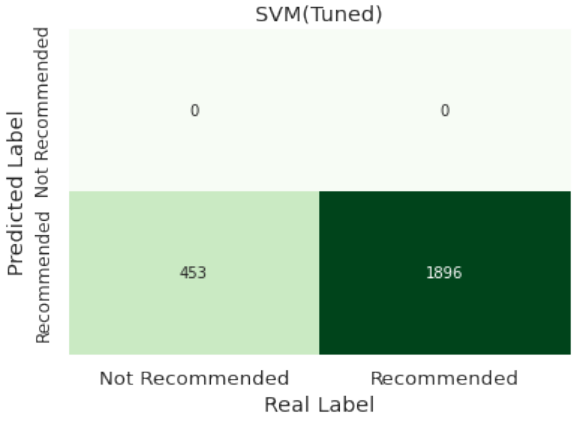
6.2.4 SVM

The hyperparameters space we search and the best results are showed below.

- C: 0.005, 0.01, 0.1, 1, 2, 5, **10**
- kernel: **poly**, rbf

Prediction and scores are as follow.



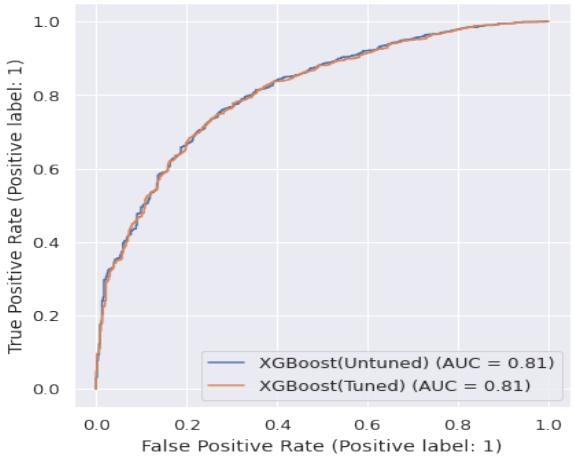


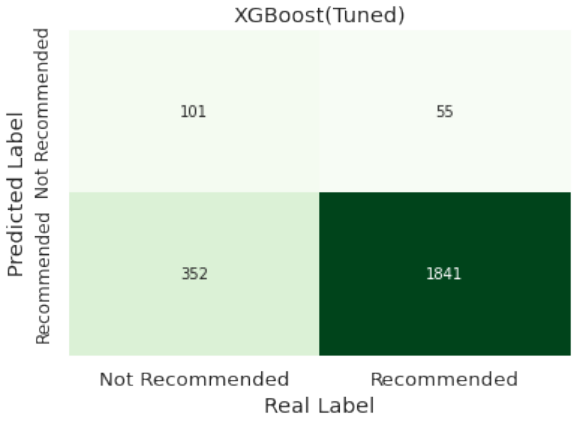
6.2.5 XGBoost

The hyperparameters space we search and the best results are showed below.

- n_estimators: 20, 50, **100**, 500
- min_child_weight: 1, 2, **5**, 10
- learning_rate: **0.05**, 0.1, 0.3

Prediction and scores are as follow.



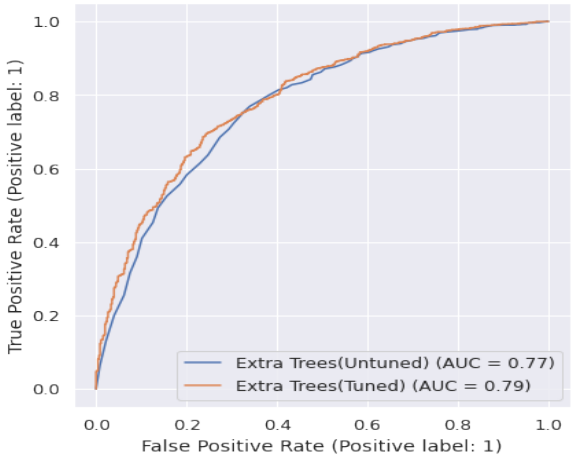


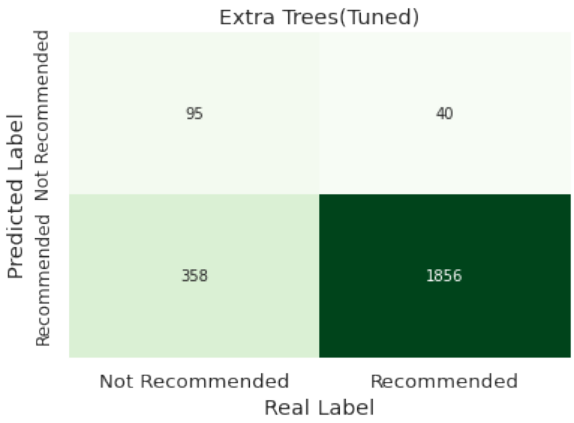
6.2.6 Extra Trees

The hyperparameters space we search and the best results are showed below.

- n_estimators: 50, 100, **200**
- max_features: **auto**, log2
- min_samples_split: 5, **10**, 20, 50

Prediction and scores are as follow.



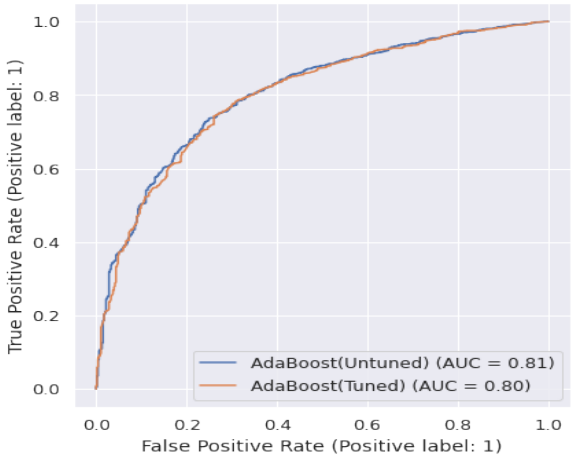


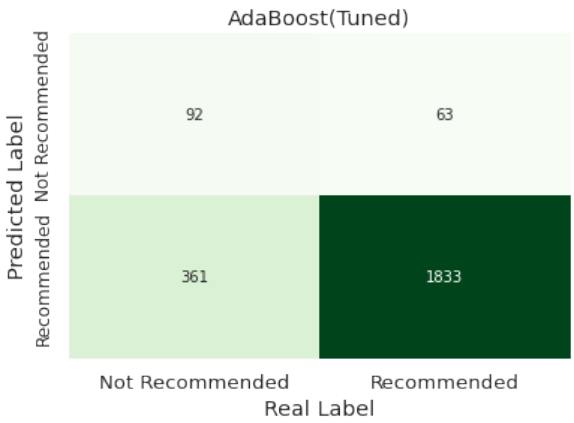
6.2.7 AdaBoost

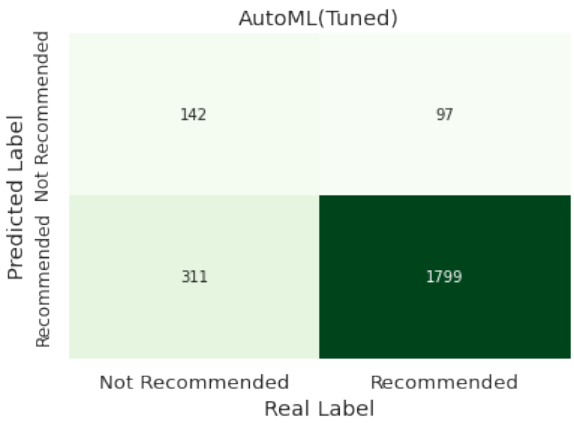
The hyperparameters space we search and the best results are showed below.

- n_estimators: 50, **100**, 500, 1000
- learning_rate: 0.01, **0.1**, 1.0

Prediction and scores are as follow.







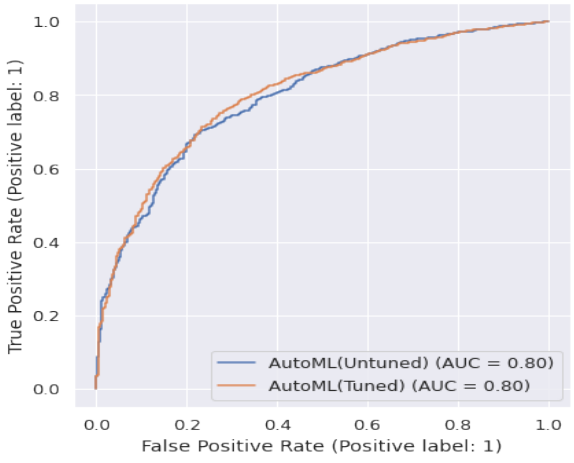
6.3 AutoML

AutoML takes a *time_left_for_this_task* as its model search time, which is not suitable for *HalvingGridSearchCV* since it can no longer save time by successive halving. So we use ordinary *GridSearchCV()* here and divide training set (90%) and validation set (10%).

AutoML has its own searching and emsembling way and doesn't leave much space for parameters tuning. We tried some hyperparameters below and find there is no difference.

- ensemble_size: 5, 10, 30, 50, 70
- ensemble_nbest: 5, 10, 30, 50, 70

Prediction and scores are as follow.



6.4 Convolution Neural Network

We use pre-trained 300-dimensional GloVe embeddings[7] to generate the word embeddings and tune the hyperparameters on the validation dataset. The CNN model has two convolution layers with the filter size set to 2, each size has 4 filters. The dropout probability was set to 0.8. We integrate metadata 'Age', 'Division Name', 'Department Name', and 'Class Name' into the CNN model by concatenating the output of the convolutional layers and the metadata vector then feed to a fully connected layer.

To reduce the influence of skewed distribution of labels, we use oversampling to make the number of zeros and ones to be the same in the training set. The loss function we use is BCEWithLogitsLoss, and the batch size for stochastic gradient descent is 8, the loss on the train and validation set is shown in Figure 5. The model is trained for 10 training epochs. The accuracy on the validation set is 82.42%.

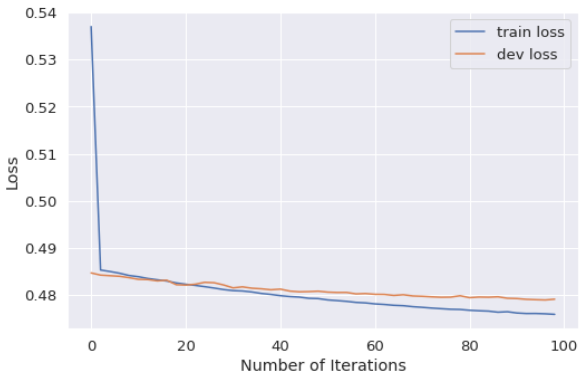


Figure 5: Train loss and validation loss during the training of CNN.

6.5 Pre-trained BERT model

We transfer the 'bert-large-uncased' model from Huggingface to our dataset, and fine-tune it to test

its performance. We utilize the BERT tokenizer to generate the Token embeddings, Segment embeddings, and Position embeddings for the review texts, and feed them into the pre-trained BERT model. We add a linear layer to get a binary prediction from the output of BERT.

During training, we let parameters of all layers in the model to be updated and use the AdamW optimizer with learning rate set to 0.001. We also create a schedule (cosine schedule with warm-up) with the above learning rate, so that the learning rate decreases gradually after certain number of training steps. After setting the batch size to 64 and training 5 epoches, then accuracy on the validation set reaches 88.38%. The confusion matrix of the prediction of this model on the test set is shown in Figure 8.

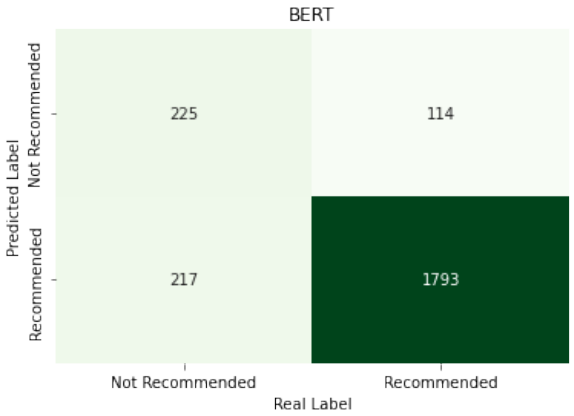


Figure 6: Confusion matrix on the test set for prediction of BERT model.

6.6 Performance Summary

The performances of each methods are showed below.

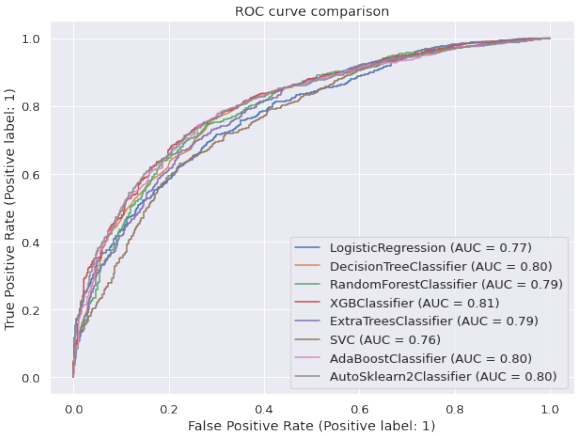


Figure 7: ROC curve comparison

Algorithm	Precision	Recall	F1	ACC
Logistic Regression	97.363	83.341	89.808	82.163
Decision Tree	96.941	82.607	89.202	81.056
Random Forest	96.677	84.587	90.229	83.099
XGBoost	96.624	84.269	90.025	82.716
Extra Trees	97.521	83.893	90.195	82.886
SVM	96.677	83.167	89.415	81.524
AdaBoost	95.200	84.385	89.467	81.907
CNN	99.631	80.692	89.167	80.460
Pre-Trained BERT	94.022	89.204	91.550	85.909

Table 2: Models Performance Comparison

7 Conclusion and Discussion

We present a project that shows modeling the real reviews of customers is important for E-commerce. Compared to most text classification study, we integrate metadata into our models and show that the model gives a highly accurate prediction when combining metadata with text. We also provide error analysis and discussion of each model's performance.

Tuning hyperparameters for traditional machine learning models doesn't have much improvements in our project, except for the decision tree. Untuned decision tree performs far worse than others, but after increasing *min_samples_split*, the performance was largely improved, almost same as random forest. We think the reason might be the original decision tree is deeply troubled by overfitting. However, when increasing *min_samples_split*, we force it to use more data to split its internal nodes, thus avoiding fitting to some outliers.

AutoML is beyond our expectation since it is so easy to use and gives first-tier results just after three minutes training, due to consideration of fairness among other traditional models. It seems there is still big space for improvement when giving more training time.

In addition, there are 38 features in our machine learning problems, and the importance of each feature to the final prediction is different. To measure how much each feature contributes to the label, we

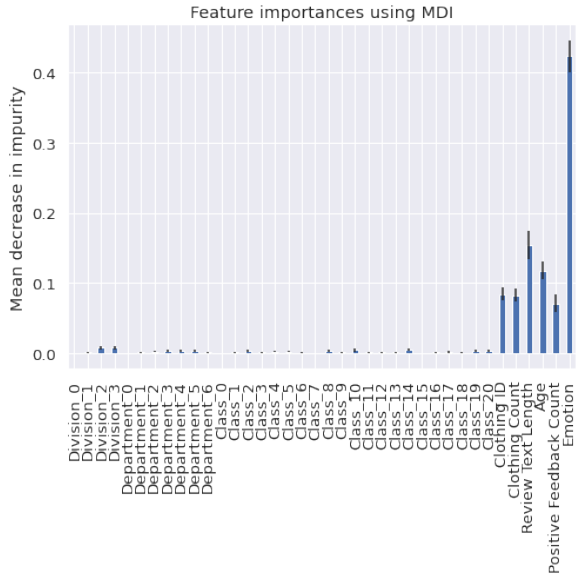


Figure 8: Random Forest feature importance based on mean decrease in impurity(MDI).

use a forest of trees to evaluate the importance of features. The impurity-based importance is shown in figure 8. The blue bars are the feature importance of the forest, along with their inter-trees variability represented by the error bars. From the figure we notice that the feature importance of last 6 features are much higher than the first 32 features, which are one-hot vectors of categorical features. Moreover, the last feature, which represents the sentiment analysis of review text, has the highest feature importance based on mean decrease in impurity. Therefore, "Review Text" will have a larger effect on the model that is being used to predict "Recommended IND".

For deep learning models, CNN seems to have trouble learning from reviews, especially with a skewed distribution, meanwhile the BERT model is easy to transfer and shows strong ability to distinguish positive and negative reviews.

As the E-commerce keeps growing, analysis of customers reviews and the building of predictive models in this area is of great significance. If given more data, our model can be applied to more generally department than woman's clothes.

References

[1] Abien Fred Agarap. *Statistical Analysis on E-Commerce Reviews, with Sentiment Classification using Bidirectional Recurrent Neural Network (RNN)*. 2020. arXiv: 1805.03687 [cs.CL].

[2] Zihang Dai et al. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. 2019. arXiv: 1901.02860 [cs.LG].

[3] Matthias Feurer et al. *Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning*. 2021. arXiv: 2007.04074 [cs.LG].

[4] Yoon Kim. *Convolutional Neural Networks for Sentence Classification*. 2014. arXiv: 1408.5882 [cs.CL].

[5] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[6] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[7] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[8] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[9] Zichao Yang et al. "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://aclanthology.org/N16-1174.