Super Learner Analysis For Classification Tasks

CS 4333 Machine Learning UAFS

Finals Pre-Report

Tony Aldana, Joe Neal

May 1, 2025

Abstract

This paper researches the use of AI systems to predict the star ratings for Amazon Reviews, using historical review text for training. Said system can be applied to any text including casual comments from bloggers or internet forum posts. Jupyter Notebooks powering Python and supported by assorted Python packages were used for development.

Four different datasets were targeted, each of varying size and subject matter. Four AI packages (Roberta, RNN, SVM via scikit-learn's SVC, and XGBoost) and a combination (XGBoost and Roberta) were explored to create AI classification models, the target being the star rating. Each had a star rating of 1 to 5. The data was cleaned up, eliminating nonsense words and unimportant words in Python.

1 Introduction

There have been a number of techniques that have been created to facilitate classification tasks. Such methods are support vector machine algorithms, extreme gradient boosting, and neural networks. However, each model has tradeoffs associated to them. This paper aims to address those tradeoffs by using an ensemble model that combines the strengths of models while also addressing their weaknesses. Such models have been shown to perform well and also have the added benefit of having the minimum performance of the best combined model. This paper focuses on using the RoBERTa model with Support Vector Machines to achieve superior results.

2 Background

Some techniques have been created that allow for the use of neural networks to address classification tasks with time series data, such as RNN [9] or using transformers to learn the relationships between text before fine-tuning with the BERT architecture in [4]. Work has also been conducted using support vector machines [3] and Gradient Boosting with XGBoost [2]. However, this paper focuses on the use of super-learners [11], a technique that uses multiple models to account for the weakness of individual models. This type of model has been used for the identification of diseases in [5] with great success. Our model focuses on achieving greater results than the individual techniques presented by combining the strengths of the RoBERTa model and Support Vector Machines.

3 Specification

3.1 RoBERTa and BERT

To first understand what RoBERTa is, the architecture of BERT must be understood. BERT stands for Bidirectional Encoder Representations from Transformers. BERT aims to train bidirectional representations of text by using the context from both the left and right directions. BERT uses the transformer architecture from [12] with the modification of being able to gain context from both sequence directions. However, to account for tokens to gain insight by being able to see itself, they mask random tokens in a sequence and create a task to predict those tokens. This allows the BERT model to gain better representation by accounting for both contexts. The second unique thing that BERT does to get the representations is sentence prediction. This is trying to predict what sentence comes after a given sentence. This gives representation the context from future sentence, which allows it to build better representations. RoBERTa follows the same architecture of BERT but has some training design changes. First the RoBERTa training procedures uses dynamic mask patterns for their token masking, full sentences without NSP loss for their sentence prediction, large mini-batches, and larger byte-level Byte Pair Encoding(BPE). RoBERTa is also trained on a significant increase in data compared to BERT.

3.2 Support Vector Machine

Support Vector Machines are a type of model that tries to plot points into an infinite dimension and find support vector classifiers to classify that data. It does so by using a kernel function. In our implementation of support vector machines we use a function known as the Radial Basis Function(RBF).

$$RBF = e^{-\gamma(a-b)^2}$$

This Function allows for this support vector machine to act like a weighted nearest neighbor function. That is, points that are closer to the observation of interest have a greater impact on how the observation is classified compared to points that are farther away. Now, in order to better optimize the kernel a term is introduced, C, which acts as a regularization term. C specifies the margin of error is allowed in a hyperplane that RBF finds. Larger values of C allow for a smaller margin of error.

3.3 Super Learner/ Ensemble Stack

Super Learners is an ensemble technique that combines a different number of learners to perform better in tasks. The proposed super-learner looks like the following:

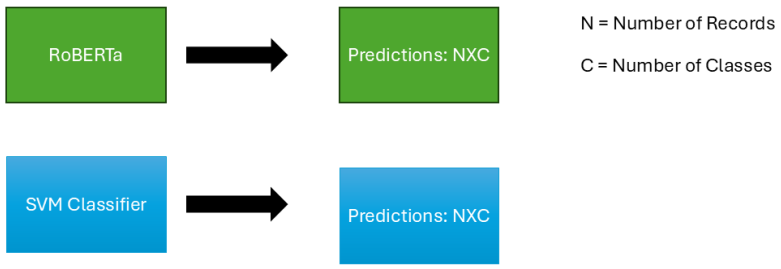


Figure 1: Shows the creation of data for the meta learner

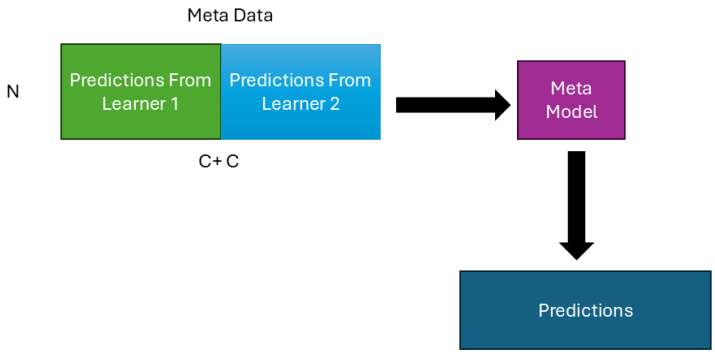


Figure 2: Shows the learners making predictions.

First the RoBERTa and SVM Models run and make their predictions. Then the two predictions from the learners are concatenated such that the resulting vector has a shape of $N \times (C + C)$. This data is then sent to the meta model to make predictions. We decided to use an SVM as our meta model.

4 Implementation

Our implementation used the Scikit-learn, PyTorch, and SKorch libraries for easy model creation. We decided to test 4 models and combine the 2 top-performing models for our stack model. We also used a Support Vector Machine as our meta-model. The 4 models that we chose to conduct experiments on were RoBERTa, an RNN, XGBoost, and an SVM. Hyperparameter tuning was conducted by grid search for the RNN, XGBoost, and SVM models. Our task was to use Amazon customer reviews to correctly predict the review rating.

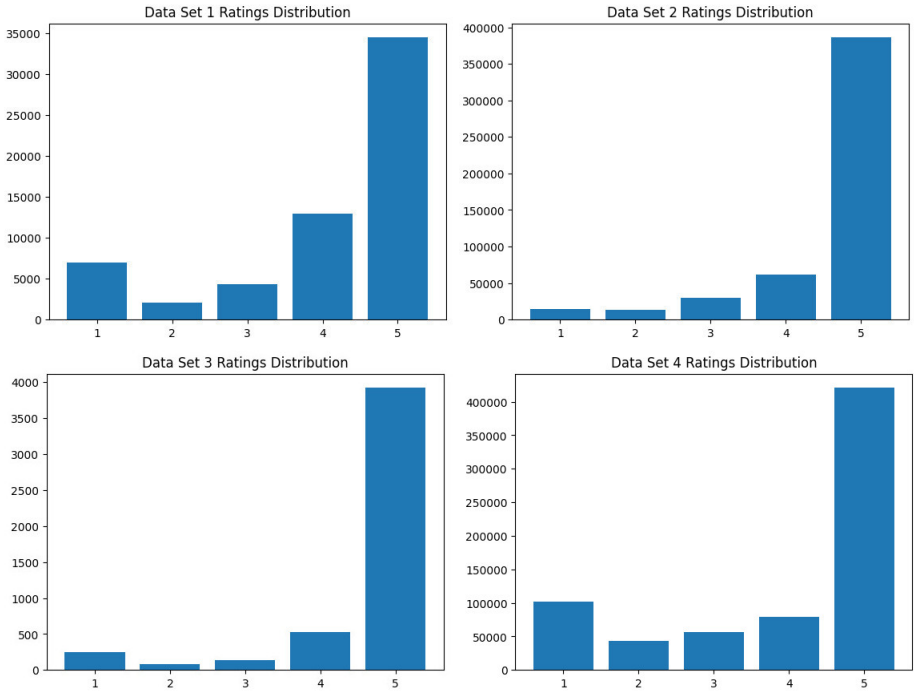
4.1 Data and Data Preprocessing

4 Amazon review data sets, were used during the training process. We used the NLTK stopwords Library to get rid of any stop words that were in the text and removed any entries that had missing data. The Following shows the review distribution for all 4 data sets.

4.2 Skewed Data

The base data was skewed towards the ends: most reviews included the far ends of low or high star ratings. The issue with this is that a person might create a system to predict a rating by creating a random number that favors low or high ratings.

4.2.1 Skewed Data - Illustrated by Graphs



Raw Numbers.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
5	34,460	385,989	3,922	420,726
4	12,976	62,170	527	79,381
3	4,366	29,495	142	56,307
2	2,108	12,954	80	43,034
1	6,979	14,599	244	102,080
Total	60,889	505,207	4,915	701,528

Rounded to significant digits.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
5	34,500	386,000	3,920	420,700
4	13,000	62,200	530	79,400
3	4,400	29,500	140	56,300
2	2,100	13,000	80	43,000
1	7,000	14,600	240	102,100
Total	61,000	505,300	4,910	701,500

Represented as a percentage.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
5	57%	76%	80%	60%
4	21%	12%	11%	11%
3	7%	6%	3%	8%
2	3%	3%	2%	6%
1	11%	3%	5%	15%
Total	100%	100%	100%	100%

4.3 Training and Testing (K-Fold Validation) and Sampling

Rather than splitting data into training and testing data, K-fold Cross Validation was used. K was defined as 5, and the data was sampled as 5000 records for dataset 1, dataset 2, and dataset 4. 2000 samples were used for dataset 3. For the Roberta Model the data was split 80/20, and k-fold was not implemented.

4.4 Hyperparameter Tuning

20 estimators were used for XGBoost, and 5 epochs were used for RNN and RoBERTa. The Sci-Kit Learn TF-IDF Vectorizer was used to make text into number representations for the XGBoost and SVM models, while the Word2Vec embeddings were used for the RNN. The pretrained RoBERTa model from Hugging Face was used and finetuned for our needs.A helper method was created for easier hyperparameter tuning. A helper method was created to do multiple grid search runs after the best parameter was returned. For XGBoost and SVM, 4 runs were conducted to tune the learning rate and lambda for XGBoost and the C constant for the SVM. XGBoost started with a value of $\lambda = .3$ and $lr = .001$ to start for all runs. $C = .7$ for all runs. For the RNN, the starting values were $lr = .0001$ and $\lambda = .5$ The following describes the algorithm that was used for hyperparameter

tuning: Hyperparameter tuning was conducted on all 4 Amazon review data sets, and the best values

Algorithm 1 Tuning Helper

while $i < runs$ **do**
  $factor = 0 < value < 1$
  $high = parmater\ * (1 + factor)$
  $low = parmater\ - (parmater * factor)$
  $A = [high, parmater, low]$
  Run Grid Search
  $paramater = gridsearch\ best\ paramater$
end while

were found. For the Roberta model the values $\lambda = .7$ and $lr = .00001$ was used for all datasets.

4.5 Data Set 1

The following hyperparameters were found for this dataset.

	lr	λ	C
XGBoost	.0004	1.1525	NA
SVM	NA	NA	3.5437
RoBERTa	.00001	.7	NA
RNN	.000156	.72	NA
SVM+RoBERTA	NA	NA	3.5437

4.6 Data Set 2

The following hyperparameters were found for this dataset.

	lr	λ	C
XGBoost	.0004	1.1525	NA
SVM	NA	NA	3.5437
RoBERTa	.00001	.7	NA
RNN	.0001	.5	NA
SVM+RoBERTA	NA	NA	3.5437

4.7 Data Set 3

The following hyperparameters were found for this dataset.

	lr	λ	C
XGBoost	.0004	1.1525	NA
SVM	NA	NA	3.5437
RoBERTa	.00001	.7	NA
RNN	.0002	.98	NA
SVM+RoBERTA	NA	NA	3.5437

4.8 Data Set 4

The following hyperparameters were found for this dataset.

	lr	λ	C
XGBoost	.0004	1.1525	NA
SVM	NA	NA	3.5437
RoBERTa	.00004	.588	NA
RNN	.0001	.5	NA
SVM+RoBERTA	NA	NA	3.5437

It was determined that the Roberta Model And SVM Model performed the best among the individual learners thus they were combined.

5 Evaluation

Five Runs were conducted, and the F1, recall, precision, and accuracy scores were recorded. The mean of those scores plus or minus the standard deviation, is what is being presented below.

5.1 Data Set 1

	Accuracy	F1	Precision	Recall
XGBoost	56.57% ± .0750%	.2000 ± .0000	.1445 ± .0012	.1131 ± .0015
SVM	55.18% ± .0940%	.3399 ± .0078	.3412 ± .0080	.3537 ± .0091
RoBERTa	64.31% ± 1.340%	.3522 ± .0439	.3519 ± .0658	.3731 ± .0241
RNN	56.93% ± .1000%	.2055 ± .0087	.1509 ± .0089	.1208 ± .0103
SVM+RoBERTA	**75.99% ± .2460%**	**.7295 ± .0543**	**.6714 ± .0561**	**.6436 ± .0544**

5.2 Data Set 2

	Accuracy	F1	Precision	Recall
XGBoost	76.17% ± .0200%	.2000 ± .0000	.1729 ± .0003	.1565 ± .0091
SVM	73.42% ± .0570%	.2314 ± .0033	.2342 ± .0064	.3119 ± .0388
RoBERTa	76.73% ± .1050%	.3502 ± .0061	.3827 ± .0179	.3388 ± .0074
RNN	75.90% ± .0440%	.2000 ± .0000	.1726 ± .0006	.1518 ± .0009
SVM+RoBERTA	**83.51% ± .2751%**	**.8402 ± .0734**	**.7071 ± .06487**	**.6513 ± .0551**

5.3 Data Set 3

	Accuracy	F1	Precision	Recall
XGBoost	79.93% ± .0980%	.2000 ± .0000	.1777 ± .0012	.1599 ± .0020
SVM	79.79% ± .0980%	.2315 ± .0111	.2312 ± .0166	.3116 ± .0297
RoBERTa	80.15% ± .1560%	.2740 ± .0099	.2523 ± .0103	.3020 ± .0158
RNN	76.59% ± .5220%	.2011 ± .0024	.1931 ± .0323	.1558 ± .0120
SVM+RoBERTA	**99.34% ± .0300%**	**.9526 ± .0395**	**.9517 ± .0357**	**.9584 ± .0288**

5.4 Data Set 4

	Accuracy	F1	Precision	Recall
XGBoost	59.78% ± .0198%	.2000 ± .0000	.1496 ± .0003	.1195 ± .0004
SVM	65.15% ± .0546%	.2314 ± .0033	.2342 ± .0064	.3119 ± .0388
RoBERTa	68.00% ± .0804%	.3894 ± .0092	.4074 ± .0274	.4001 ± .0116
RNN	48.14% ± .8450%	.2010 ± .0020	.1422 ± .0203	.1409 ± .0408
SVM+RoBERTA	**83.15% ± .2751%**	**.8402 ± .0734**	**.7071 ± .0649**	**.6513 ± .0551**

6 Conclusions

The Super Learner model greatly outperformed the individual learners in all categories. This further proves the effectiveness of the ensemble learning. However, more test should be conducted to ensure that the model is not overfitting the data. Such test might be more closely analyzing the results per class and using more samples for each experiments.

References

[1] Christopher J.C. Burges. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, August 2016.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. October 2018.

[5] Alireza Ghasemieh, Alston Lloyed, Parsa Bahrami, Pooyan Vajar, and Rasha Kashef. A novel machine learning model with stacking ensemble learner for predicting emergency readmission of heart-disease patients. *Decision Analytics Journal*, 7:100242, June 2023.

[6] Michael I. Jordan. *Serial Order: A Parallel Distributed Processing Approach*, pages 471–495. Elsevier, 1997.

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. July 2019.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. January 2013.

[9] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. November 2019.

[10] Karl Thurnhofer-Hemsi, Ezequiel López-Rubio, Miguel A. Molina-Cabello, and Kayvan Najarian. Radial basis function kernel optimization for support vector machine classifiers. July 2020.

[11] Mark J. van der Laan, Eric C Polley, and Alan E. Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), January 2007.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. June 2017.

[13] Steven Young, Tamer Abdou, and Ayse Bener. Deep super learner: A deep ensemble for classification problems. March 2018.