# Machine Learning & Deep Learning Approaches to Predict Student Attitudes from RateMyProfessors Comments

Yuanxin (Zoe) Zhang (yz6201),
Yumeng (Skyler) Chen (yc4144)

***Abstract*—Recent years have seen a surge in online review platforms. Using 19,005 student comments from RateMyProfessors, along with relevant user and course features, this project aims to train appropriate machine learning models for predicting reviewers' attitudes. In this paper, we will first introduce our dataset and data pre-processing method, then we compare the performance of different machine learning models we apply, including Logistic Regression, Decision Trees with Adaptive Boosting, Support Vector Machines, and Convolutional Neural Networks.**

**We use AUROC and accuracy as the two main measures of our model performance. Based on our model results, the best model we developed is logistic regression with the 'newton-cg' solver with L2 norm penalty term and regularization strength C=1, which leads to an AUROC score of 0.661 and an accuracy rate of 74.4%. We discuss the limitations of our project and also point out some future directions and possible improvement.**

## I. Introduction

The emergence of online course evaluation platforms such as RateMyProfessors has drastically altered the way students acquire course-related information. Students can submit ratings of professors' quality of teaching on this platform, which is often taken into account by future students before deciding which courses to enroll in. Professors may also find the ratings and feedback useful in adapting their teaching methods.

Despite the fact that reviews on RateMyProfessors are an invaluable source of information that could provide insight into education choices and improvements, only limited research has been conducted to analyze and predict attitudes expressed on RateMyProfessors, which becomes the focus of our project.

We mainly focus on predicting students' attitudes from their comments, which, we believe, provide in-depth information about how students view professors' teaching abilities and techniques. In addition, since user- or course-related factors, such as the number of students and the difficulty of the course, may also be crucial in predicting students' attitudes, we also include them in our model.

Essentially, our project goal is to solve a classification problem, where the predicted outcome is whether the student's attitude is 'positive' or 'negative', with the label generated as 'positive' if the actual rating is greater than 3, and 'negative' otherwise. Considering that comments and course features possess imbalanced lengths and distributions, we expect to train a robust machine-learning classifier capable of predicting most cases with relatively high accuracy.

## II. Dataset

The data given is a *csv* document that contains professor_name,school_name, department_name, local_name, state_name, num_student, student_star, student_difficult, attendance, grades, comments, etc. There are 55 various features and more than 20000 comments in total. To abstract the most effective features which would finally influence the classification of labels, we choose the most important 6 features, department_name, state_name,num_student, attitude(label), student_difficult,word_comment, and comments. The department_name and state_name are transformed into numbers of categories without other normalization. And we normalized the columns of num_student, student_difficult, and word_count since they have a huge difference which might cause the final result to be invalid. So here we need to conduct normalization for these three features. Attitude(label) is either 1(positive) or 0(negative), and the benchmark of attitude depends on the column of student_stars. For every comment whose rate is larger than 3, it's positive; otherwise, it's negative. Each comment is a sentence or several sentences enclosed in " ".

The comments are a mixture of words, numbers, URLs, and references to people. Words contribute to predicting the attitude, but numbers, URLs, and references don’t. Meanwhile, we should notice the extra punctuation, misspelled words, and some repeated characters in a word to express attitudes. Therefore, we need to preprocess the comments.

## III. Methodology and Implementation

### A. Pre-processing

Raw comments from RateMyProfessors would result in a noisy dataset. Students who commented on the RateMyProfessors website might have a casual language habit. Therefore, raw comments should be preprocessed to create a dataset to be easily learned by various classifiers. Here are some standardized steps to reduce the raw data size.

- Convert all comments into lowercase. (notice that the function lowercase() could not recognize numbers.)
- Replace all dots with space.
- Strip space and quotes(” and ’)at the end of each comment
- Replace two or more spaces with one single space.
- Replace all URLs with the word ”URL”. (The regular expression used to match URLs is www[˙§]+)—(https?://[§]+))
- Replace all hashtags with the words with the hash symbol. (For instance, #perfect becomes perfect. The regular expression used to match hashtags is #(§+))

After applying the comment’s level of preprocessing, we proceed with individual words in each sentence.

- Remove all the punctuation [’”?!,.():;] from words
- Convert more than two letter repetitions to 2 letters. (For instance, some students might write comments like *She’s soooooo gooooood* to strengthen their attitude. To continue the word-embedding part afterward, we transform this kind of comment into *She’s so good.*
- Remove - and ’. Some words might have these signs as part of the words, for instance, t-shirt, there’s, etc. We convert them into tshirt and theres to help the ensuing process.

### B. Word-Embedding

Word Embedding is a good transformation from textual data into vectors or matrices so that we can use these data for the ensuing training. In this case, we have to satisfy some principles. 1) Every word in one sentence should have a relation, which means every word is not an individual number, but in correlation with the previous and ensuing words or sentences. 2) Since we regard one comment as one input, therefore, regardless of the length of the comment, we should transform them into a uniform-length vector. Compared to various sentence transformers such as Word2Vec, and Bert, we observed that Bert might have a higher accuracy from texts to vectors. We utilize the sentence transformer all-MiniLM-L6-v2 model from Hugging Face. This model maps sentences to a 384-dimensional dense vector space.

Therefore, after the data preprocessing and word embedding process, we get a matrix $19005 * 384$. Meanwhile, considering the previous features other than comments, we ought to have a $19005 * 389$ matrix.

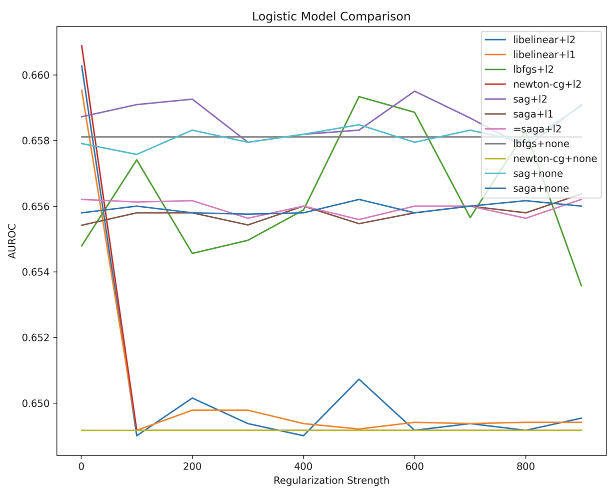
## IV. Models

We mainly use the AUROC score to compare different model performances, which is one of the major matrices to evaluate classification models in particular. We choose the AUROC measure because it is a relatively comprehensive measure that could capture both the True Positive Ratio (TPR) and False Positive Ratio (FPR) under the entire Receiver Operating Characteristic (ROC) curve, yielding an aggregate measure of classification model performance. In addition, we also present the accuracy and confusion matrices of using the test dataset as auxiliary tools to illustrate the model performance.

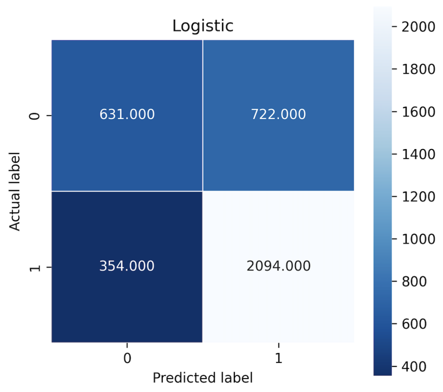
### A. Logistic Regression

We first apply logistic regression, which would be a suitable model for our case, since our predicted output Attitude (either 0 or 1) is binary. For logistic regression, the main hyper-parameters we may tune in scikit-learn are solver, penalty, and regularization strength. It is worth mentioning that by tuning the model’s complexity, regularization finds a good bias-variance tradeoff. We tried different solvers including ’lbfgs’, ’liblinear’, and ’newton-cg’, and we also try without penalty and penalty terms with L1 and/or L2 norm, depending on what the solver accepts. In addition, we apply grid search to tune the regularization strength to find the hyper-parameter with the most optimal performance.

We have created a graph for visualization that illustrates the model performance with different solvers, regularization strengths, and penalties we have tried, characterized by AUROC scores. We can see that among all the hyper-parameters we tried, the ’newton-cg’ solver with L2 norm penalty term and regularization strength C=1 yields the best result, with an AUROC score of 0.661 and an accuracy of 74.4%, using the testing dataset.

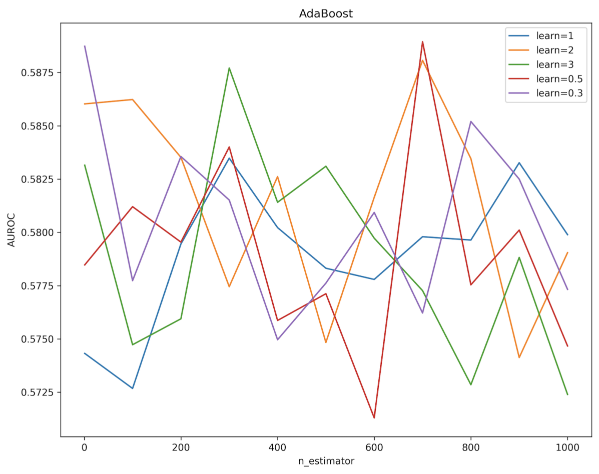


The confusion matrix for the logistic model with the best AUROC score is presented below. In the confusion matrix, specifically, diagonal elements (in our case, 631 and 2094) correspond to points whose predicted label is equal to the true label, whereas off-diagonal (in our case, 354 and 722) elements correspond to those mislabeled by the classifier. A higher confusion matrix diagonal indicates fewer wrong predictions, meaning the model is more accurate.

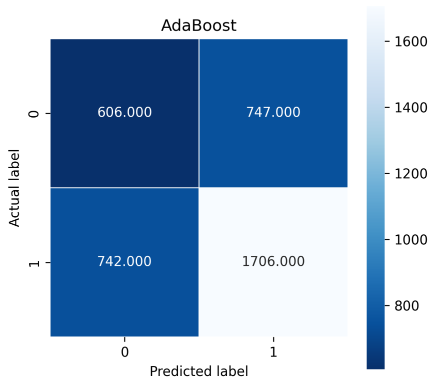


## B. Decision Tree with Adaptive Boosting

We then turn to Adaptive Boosting, which involves sequentially generating decision trees and learning from previous models. This involves punishing incorrectly predicted sample data by assigning higher weights to successor models. We mainly tune two hyper-parameters, including n_estimators and learning rate. Using grid search, we found the best hyper-parameters for the AdaBoost model are learning rate=0.3 and n_estimators=700. Our test set yields an AUROC score of 0.589 and an accuracy of 70.0%.



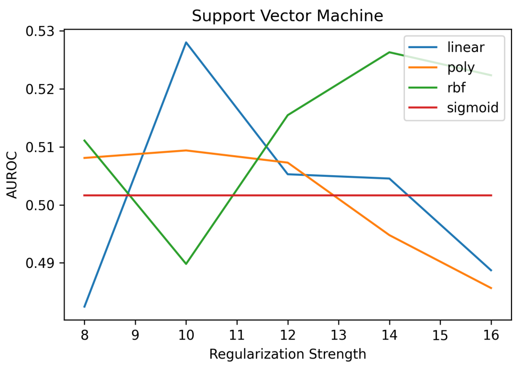
We also attach below the confusion matrix of the best-performance AdaBoost model (learning rate=0.3 and n_estimators=700). As we can see, the diagonal elements (606 and 1706) are smaller than those in the logistic regression case, indicating a worse classification performance since the model has fewer correct classifications. This is consistent with what the AUROC and accuracy scores indicate, which are both lower than those of the logistic regression too. This means that the AdaBoost model has a relatively worse performance compared with the logistic model that we presented in the last section.



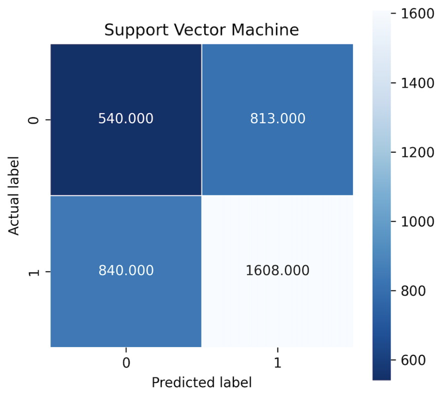
## C. Support Vector Machine

In this section, we present the results we get after applying the Support Vector Machine. In our case, SVM is suitable for our classification task since it maximizes the width of the gap between the ’positive’ and ’negative’ categories, i.e. the margin, and it also has the advantage of performing non-linear classification efficiently through the use of kernel trick and mapping inputs into high-dimensional feature spaces.

We present the AUROC score obtained after applying different kernels with different regularization strengths. We found that the model with the best performance as measured by the AUROC score is using a linear kernel with C=10, which has an AUROC score of 0.528, and an accuracy of 66.4%.

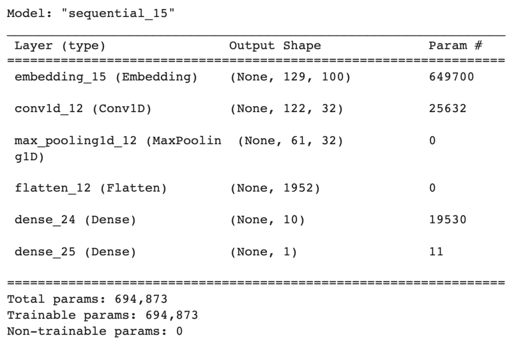


Lastly, we present the confusion matrix for the Support Vector Machine with the best performance.



## D. Convolutional Neural Network

Meanwhile, we use **Keras** with **Tensorflow** to implement a simple convolutional neural network to train the dataset. Here we use a different word-embedding skill first, then construct CNN to train the dataset and predict the test dataset. For the word-embedding part, we build up a dictionary that covers all words in all comments. Then we filter the words whose occurrences are smaller than two. And then we tokenize and encode the previous data to construct the input dictionary and input matrix. After we got the input data, we construct a CNN shown in the following figure.



```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_15 (Embedding)    (None, 129, 100)          649700

 conv1d_12 (Conv1D)          (None, 122, 32)           25632

 max_pooling1d_12 (MaxPoolin  (None, 61, 32)           0
 g1D)

 flatten_12 (Flatten)        (None, 1952)              0

 dense_24 (Dense)            (None, 10)                19530

 dense_25 (Dense)            (None, 1)                 11

=================================================================
Total params: 694,873
Trainable params: 694,873
Non-trainable params: 0
```

After we build up the model, we train the dataset with 10 epochs and get a training accuracy 98% and model loss 7%. And After I tuned the learning rate and epochs, I found the optimal test accuracy is around 60% with a learning rate of 0.01 and epochs 10.

## V. Discussion and Future Work

### A. Disucssion

We have created a table to conclude our model results:

| | Logistic | AdaBoost | SVM | CNN |
|---|---|---|---|---|
| AUROC | 0.661 | 0.589 | 0.528 | / |
| Accuracy | 74.4% | 70.0% | 66.4% | 60% |

The best model we developed is logistic regression with the 'newton-cg' solver with L2 norm penalty term and regularization strength C=1, which leads to an AUROC score of 0.661 and an accuracy rate of 74.4%. However, this accuracy is still not very satisfactory, and our project has several limitations. We also would like to point out some future directions in the following section.

### B. Limitations and Future Work

*1) Logistic Regression, Decision Tree with AdaBoost, and SVM:* Due to the high dimensionality nature of our dataset, for each hyper-parameters we tuned, it took relatively long periods to get the resulting scores. Therefore, the accuracy of our models might be low because of the limited number of grids we tried during the grid search process. To improve the performance of our models, future work should be applied to conduct a more well-rounded tuning process that involves trying more numbers and finding the more optimal hyper-parameters. PCA might also be applied to reduce our dimensionality. Besides, to get a better sense of the performance of our classification models, although AUROC is a relatively comprehensive evaluation method, it is suggested that we also separately look at both the precisions and recall scores.

*2) Convolutional Neural Network:* Due to the Covid situation, we haven't tuned other hyper-parameters such as the dimensions of kernels, the depth of the neural network, etc. In the ensuing process, if we want to improve the accuracy, we ought to consider these hyperparameters and use cross entropy to increase the model's accuracy. Meanwhile, our model divides the attitudes into positive and negative in terms of students' stars. However, if we want to evaluate the attitude in the comments more precisely, we ought to divide them into three categories or more and implement CNN again using softmax to see the performance. We can also change the layer, for instance, change Conv1D to Conv2D, or Conv3D. In addition, considering the correlation between each word, we can use LSTM, which is one of the Recurrent Neural Networks, or combine CNN with LSTM to avoid gradient vanishing or exploding.

## VI. Conclusions

In conclusion, we tried different models for predicting student attitudes from RateMyProfessors Reviews. We first introduced our dataset and preprocessing methods. Then we presented our attempts and results of our models, including Logistic Regression, Decision Tree with Adaptive Boosting, Support Vector Machine, and Convolutional Neural Network. The best model we developed is logistic regression with the 'newton-cg' solver with L2 norm penalty term and regularization strength C=1, which leads to an AUROC score of 0.661 and an accuracy rate of 74.4%. In the future, more effort should be made to better tune hyper-parameters in our models. More complicated models should also be applied to potentially improve the performance. It is also suggested to divide attitudes into more categories than the simple binary one that we implemented.