CS4372 Project 2 Report

Salma Khalfallah SMK210009
Hana Al-Jaser HZA220000

October 10, 2025

1 Introduction

A decision tree is a simple flow-chart like model that generates a final decision for a given question from a dataset by asking a series of questions and partitioning the data accordingly until a final "decision" is made. Although a simple model, this is a powerful non-parametric, supervised learning method useful for both classification and regression. This model has simple overhead, powerful inference capability, and easy-to-visualize/interpret results. One useful application of decision trees is through sentiment analysis. Sentiment analysis is the process of analyzing large amounts of text data to train a model on how to gauge the sentiment of inputted text into a model. This is useful in gauging sentiment for business, marketing, social research, and more.

In this paper, we plan on utilizing various tree models such Decision trees as well as more complex models such as Random Forests, AdaBoost, and XGBoost. Our goal is to be able to build a model that is relatively successful in identifying sentiment from human text. This paper utilizes the *Sentiment Labelled Sentences* dataset on UC Irvine's Machine Learning Repository. The dataset consists of 500 negative and positively labeled sentences from Yelp, Amazon, and IMDB for a total of 3000 sentences total.

2 Pre-Processing and Data Exploration

It is important that the data is best prepared for model fitting. In order to do this, the data was first scrubbed to identify any issues in the data: missing values, duplicate rows, and null values. No missing or null values were identified in the data; however 17 duplicate rows were identified and taken care of accordingly through removal.

Next, the data was explored in order to identify potential relationships and things to note when performing modeling. Firstly, we identified the class distribution of the target class *sentiment*. This was done in order to gauge any potential caveats in the distribution when analyzing the accuracy, precision, and

more. This is especially significant in datasets with skewed target distributions, such as health data. In the case of the model, the data was not overall skewed to one specific class, but rather evenly split across the two classes *negative* and *positive* respectively.

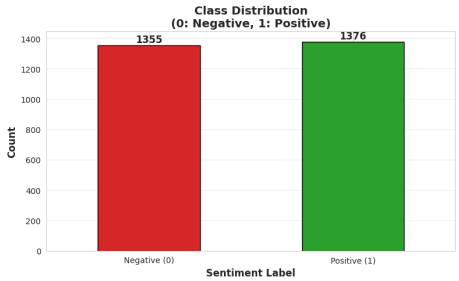

Figure 1: Class distribution of target class *sentiment*

Next, feature extraction using the *TD-IDF* measure was performed. The Term Frequency-Inverse Document Frequency value is a measure of any word's importance in a document relative to how frequently that word is encountered in day-to-day language. The equation consists of two parts: the TF and IDF values. The TF value measures how often a word appears in any one document, while the IDF value measures the weight of words across all documents, increasing the weight of rarer words. The idea is that, when a rarer word (for example, a proper noun) appears, it is more significant to a document while a more common term (for example, a conjunction) has a lower weight and is not considered as significant of a word in any given document.

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D) =$$

$$[(\frac{Count(\text{Term t appears in document } d)}{Sum(t) \forall t \in d}) \times (log(\frac{Count(\text{Documents d in corpus } D)}{Count(\text{documents d containing } t)}))]$$

The text in the datasets were extracted and converted to $TF-IDF$ features, and then merged into a data frame for easier data manipulation and analysis. There were 500 total features, with a mean $TF - IDF$ value of 0.0032 and a maximum value of 1. For example, given a sample sentence negative movie review: "A very, very, very slow-moving, aimless movie about a distressed, drifting young man.", the top $TD - IDF$ features for the sentence were: 'man', 'slow', and 'movie' at $0.6639, 0.6188$, and 0.4199 respectively.

After vectorizing the text into a feature space, it seemed only natural to analyze the relationship between the features next. The dataset was analyzed to identify the top 20 words most strongly correlated with our target *sentiment* variable. While performing the analysis, it is important to identify a relationship: a positive correlation related to a positive sentiment while a negative correlation relates to a negative sentiment in the text. The top 20 words were identified

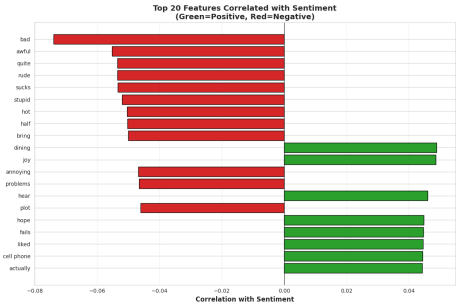

Figure 2: Top 20 Correlated words with *sentiment*

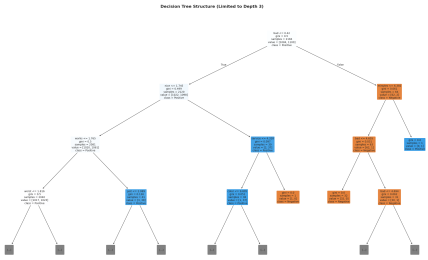

Figure 3: Decision Tree Visualization

and graphed below. (see: Figure 2) Twelve of the twenty words had a negative sentiment, while the remaining eight had a positive sentiment. The words with a negative sentiment seem to also correlate more strongly with sentiment, potentially suggesting that a negative sentiment in text is generally a more strongly worded sentence to a positive sentiment text. (note to self: please edit this part)

To conclude data pre-processing and exploration, the dataset went through feature reduction to simplify the dataset as well as data splitting (80% training, 20% testing data) and feature standardization for general data quality, accuracy, and reliability. The data is thus sufficiently prepared to go through model construction and evaluation in order to reliably model sentiment in text through various tree models.

3 Model Construction

The first model to be evaluated is a basic decision tree classifier. In addition to general modeling, hyperparameter tuning was performed on the maximum tree depth, minimum sample split, minimum samples in leaf nodes, and finally splitting criteria. This was performed in order to minimize model overfitting, reduce model complexity, and obtain greater performance from the final tree model. The best tree model was split using the Gini Index with a maximum

depth of 20, 1 minimum sample in the leaf nodes, and 20 minimum samples to split an internal node. The test set obtained a 59% accuracy, 56% precision, 96% recall, and finally an F-1 score of 70%. An interesting point to ascertain is the model's recall metric. Recalling that the recall metric is the proportion of positive class elements that were correctly identified by the model, the decision tree was excellent in identifying positive sentiment text data. However, the precision metric notably drops to 59%. This suggests that the false negatives were much higher than false positives during the testing stage. This is all reflected in the F-1 score at 70%. Although not a bad metric, there is room for improvement.

The next model to be evaluated is a random forest classifier. This classifier emerged as a response to the potential overfitting of a classic decision tree model. By implementing an additive model framework of averaging resampled model outputs, the overall variance of the forest of trees decreases as opposed to the variance of a single decision tree output. As was with the decision tree classifier model, the hyperparameters were also tuned and optimized to the dataset. The final Random Forest model had no maximum depth limit, 2 minimum sample nodes to split internal nodes, and 200 fitted decision trees. The final test set performed with accuracy 65%, precision of 61%, recall of 84%, and finally an F-1 score of 70%. The random forest improved from the decision tree classifier by 0.44%. Although not a high magnitude, the classification evaluation metrics suggest a more balanced model than before. The overall accuracy and precision of the random forest model has increased from the previous model, however the recall metric has decreased down to 84%. Although there is still room for improvement, the dataset is better fit to our random forest model.

The third model to evaluate the data with is the AdaBoost classifier model. As with the last two models, hyperparameters were tuned: number of weak learners and the learning rate. One thing to keep in mind with the AdaBoost model is that the sequential nature of the boosting algorithm means that the model can be slow relative to other tree models. This proved to be true, with the model once running for 15 minutes upon running compared to the < 30 runtime for the other algorithms in the evaluation. The final AdaBoost algorithm had an optimized learning rate of 1 and 200 weak classifiers, with an F-1 score of 70%. The model also had a final accuracy of 59%, a precision score of 55%, a recall metric of 97%. This model performed comparatively to the basic decision tree classifier from earlier, but much slower computationally. This model also did not improve from the previous random forest model.

The final model to evaluate is the XGBoost model. This is a powerful boosting model with a foundational objective function of $Obj = Loss + Regularization$. This model both minimizes the loss function to build a predictive model *and* the regularization function, building a simpler model to avoid overfitting trees. As with the previous models, hyperparameter tuning was performed with the number of boosting rounds, the maximum tree depth, learning rate, subsample ratio of training data, and subsample ratio of features. The final model ob-

tained 300 iterations, a maximum depth of 3 per iteration, a learning rate of 0.1, a subsample ratio of 1 for training data, and finally a subsample ratio of 0.8 for features. The model performed relatively well with an accuracy of 65%, precision of 60%, a recall metric of 91%, and finally an F-1 score of 73%. The XGBoost model seems to somehow balance both the metrics from the decision tree and AdaBoost classifiers as well as the Random Forest classifier. This has proven to be one of the more strong yet balanced classifiers out of the four tested.

3.1 Result Analysis

While performing analysis of the results, one significant pattern that was noted was that, despite hyperparameter tuning, the models struggled to not overfit to the training datasets. This was especially evident in models in which overfitting is not regulated as strictly, such as the classic Decision Tree classifier and the AdaBoost algorithms. This was especially noticeable when comparing the training and test data set performances for the models, visualized below. Models with stricter punishments on overfitting, on the other hand, saw stronger overall performance.

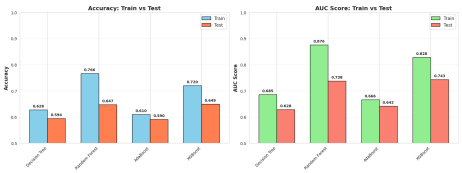

Figure 4: Training and Testing Accuracy

Another interesting pattern to note is how all four models responded to a new positive sentiment text string in the test dataset. The model responded relatively well to new negative sentiment text data, but struggled to identify positive sentiment text data. While performing feature analysis beforehand, it has also been established that the top words most strongly correlated with sentiment were all negative words. (e.g. "bad", "awful", "sucks")

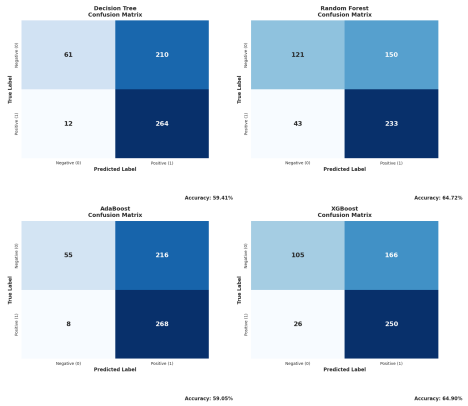

Figure 5: Confusion Matrices for Tree Models

This phenomenon as well as the previously mentioned overfitting impacted models negatively (see: Figure below)

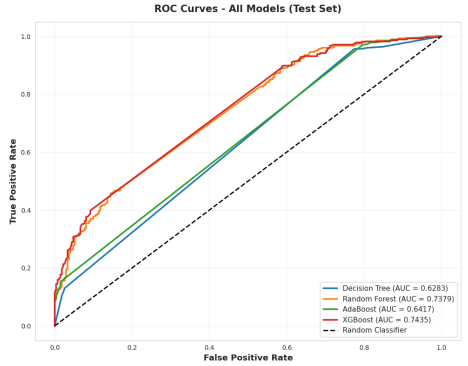

Figure 6: ROC Curves for all four models

The decision tree and AdaBoost classifiers performed as weak classifiers, struggling to barely perform better than chance. The Random Forest and XGBoost models, on the other hand, performed relatively well.

After comparing all of the models, we would recommend the XGBoost model. The model provided faster computing time, greater performance gains, and addressed overfitting concerns from previous models.

4 Conclusion

In conclusion, this report analyzed the performance of various tree and ensembling models when performing sentiment analysis. Various pre-processing and

tuning techniques were performed to prepare the data accordingly and as accurately as can be. Post-modeling revealed overfitting concerns as well as a different in performance between the classes for all the models. After careful consideration, the XGBoost model was chosen as the best model of the four given the problem; the model had greater performance gains and addressed previous model concerns. For future research, this problem could be extended to more datasets to address the overfitting concerns. As well as this, the phenomenon of the models identifying a skewed proportion of class data is a significant concern that should be addressed in any future work.