COMP 90051 Assignment 1

Group 17: Potato

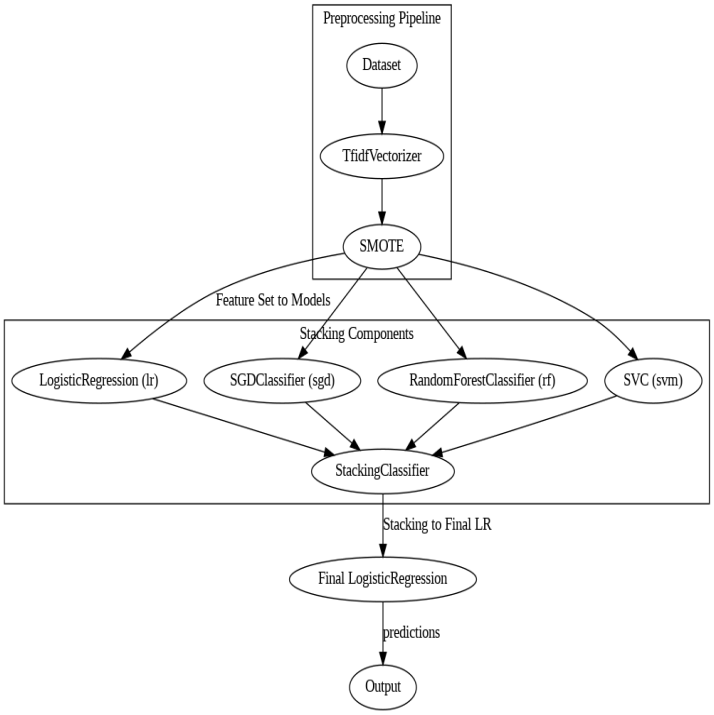
Group Members:

1. Advait Patwardhan (1398875)
2. Divyanshu Mishra (1281413)
3. Kang Qi Goh (1429170)

Project Overview

Our project involves classifying text instances as either human-generated or machine-generated given training data from two domains that contain preprocessed text data represented numerically. Our challenge is to develop a model that effectively predicts text samples in a test set if they are human or machine-generated. To address this classification task, we plan to explore several machine learning models such as Logistic Regression (LR), Support Vector Machine (SVM), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Random Forest (RF), and 1d CNN. Besides, given the imbalance of data samples in domain 2, we experimented with a few techniques to create a more balanced training environment, which includes SMOTE (Synthetic Minority Over-sampling Technique), undersampling, oversampling, and Borderline Smote (2 kinds: Borderline-1, Borderline-2)

Description of our Final Approach - Stacking Classifier



Explanation:

The entire pipeline starts by transforming raw text into TF-IDF vectors, which are then used as input for the stacking classifier. The stacking classifier processes these inputs through its base classifiers and combines their outputs through a final logistic regression model to produce the ultimate prediction.

The architecture illustrated in Figure 1 provides a visual representation of the final approach adopted in our text classification pipeline, offering a comprehensive overview of the systematic process employed to transform raw text data into actionable insights.

Figure 1: Architecture Flow Chart of Stacking Classifier

We combined datasets from two distinct domains to ensure a diverse representation and created a unified feature space. For vectorization, we used the TF-IDF approach with the following settings:

- max_df=0.50: Ignores terms that appear in more than 50% of the documents.

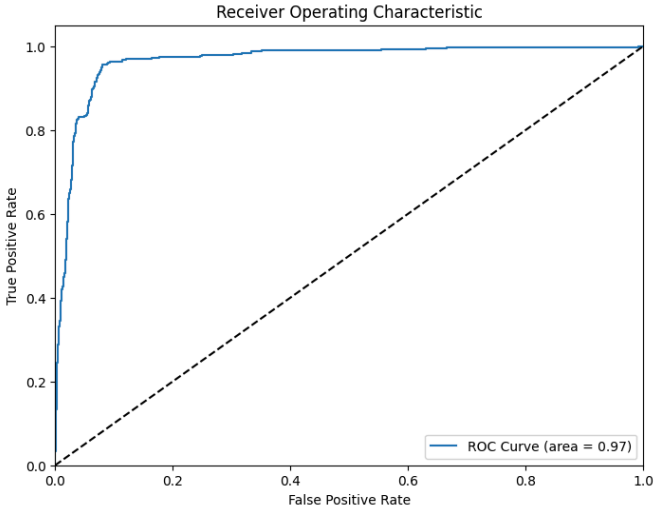
- min_df=2: Excludes terms that appear in fewer than two documents.
- ngram_range=(1,7): Captures unigrams to septigrams, enhancing context and semantic interpretation.

To address class imbalance, we employed SMOTE, generating synthetic examples in the minority class, and effectively balancing the datasets before model training. Our base classifiers with optimized hyperparameters include:

- **Logistic Regression** (regularization C=10, max iterations=1000, solver='liblinear', class weight balanced for dealing with class imbalance)
- **SGD Classifier** (max iterations=5000, using modified Huber loss, class weight balanced)
- **Random Forest Classifier** (100 estimators, class weight balanced)
- **SVC** (C=10, RBF kernel, probability estimates enabled, class weight balanced)

We optimized these classifiers using GridSearchCV with 5-fold cross-validation to determine the best hyperparameters. Finally, we stacked these models, using a Logistic Regression as the meta-classifier, to improve predictive performance by leveraging the strengths of individual classifiers.

Result:



This shows our final model performance:

ROC: 97.2%
Kaggle: 86%
Classifier Accuracy: 87.2%

Figure 2: Stacking Classifier ROC AUC

Alternatives Considered

Models were trained on training data and evaluated using six performance metrics (accuracy, precision, recall, F1, F1 macro score, and Kaggle accuracy). Results are tabulated below, supporting evidence is provided in the ‘Code’ folder:

Model	Accuracy	Precision	Recall	F1	F1 macro	Kaggle Accuracy (%)
LSTM	0.75	0.72	0.72	0.71	0.71	72.20
1D CNN	0.78	0.75	0.74	0.75	0.74	75.00
Random Forest	0.80	0.75	0.75	0.74	0.73	75.05
LSTM (bi-directional)	0.82	0.75	0.74	0.76	0.73	75.10
SVM	0.83	0.77	0.78	0.76	0.78	78.20
LR	0.85	0.86	0.85	0.82	0.70	82.40
SGD + LR	0.88	0.86	0.85	0.84	0.86	85.20
CNN + SVM+ LR	0.79	0.71	0.71	0.69	0.70	71.55
LR + SGD + RF + SVM	0.86	0.87	0.87	0.86	0.86	86.00

Discussion

We have initially attempted individual algorithm approaches but our accuracy was below expectation. Thus, we experimented with the stacking of multiple models. Stacking is an ensemble learning technique that enhances model performance by combining predictions from multiple base classifiers. Each classifier's strengths are leveraged, while their weaknesses are mitigated. In addition, we also use hyperparameter tuning to address the downside of each alternative method. Hyperparameter tuning using GridSearchCV was used to find the best hyperparameters for our models to overcome overfitting and underfitting with cross-validation, CV = 5 to increase the overall accuracy of our models.

Another challenge we faced was class imbalance. Class imbalance can substantially affect model performance. Initially, our dataset in domain 2 was imbalanced, which significantly skewed the performance of our models. Models trained on imbalanced datasets may exhibit a preference for the majority class, resulting in misleadingly high accuracy but poor recall, precision, and F1 scores for the minority class. Such imbalance often leads to biased decision boundaries and potential overfitting to the majority class features, neglecting the minority class.

To address this, we first concatenated domain 1(balanced) and domain 2 (imbalanced) training data and converted it to string form from integers. We then applied TF-IDF to vectorize the data['text'] field because models can understand it in vectorized form more efficiently. Next, SMOTE (Synthetic Minority Over-sampling Technique) was applied, synthesizing samples for the minority class. This technique generates synthetic samples from the minority class to ensure that the classifiers do not bias towards the majority class. For instance, in our dataset with 14,000 instances of label '0' and 4,000 of label '1', SMOTE was used to equally balance both label '0' and '1'.