[Gabriel Bermudez, 3965-6965]

CIS4930 Individual Coding Assignment 2

Spring 2023

1. Problem Statement

[The first step in any machine learning project is defining your problem. Here you need to present a high-level problem statement, which may include a description of the problem, why this problem needs to be solved, and how did you solve it.]

The problem consisted of building a sentiment classification model that determines the tone of a paragraph (positive or negative). The models were trained on a huge dataset containing labeled user online discussions. To extract the linguistic features from the texts, a handful of statistical and semantic feature extraction algorithms were employed, namely bag-of-words, tf*idf, and word2vec. Each of which was then used to create classifier models using the logistic regression, naive bayes, and random forest approaches. Classification reports and confusion matrices served as the main ways to evaluate and compare model performance.

2. Data Preparation

[Preparing data for training machine learning models is a fundamental step. Describe your data preparation process in detail, which may include how did you clean the data and how did you extract features from the data for the following model training steps.]

Exploratory Data Analysis steps:

A. Check the size of the datasets.
B. Check the distribution of the [‘Sentiment’] column (may need to rebalance later).
C. Check for missing values and replace as necessary.

Text Preprocessing Steps:

A. Word tokenization [word_tokenize() from nltk.tokenize]
B. Lower_casing [lower()]
C. Remove numbers [re.sub()]
D. De-contraction [re.sub()]
E. Remove punctuation, special characters, and symbols [re.sub()]
F. Stemming [SnowballStemmer(“english”) from nltk.stem.snowball]

Additional Steps:

A. Generate randomly sorted subset of dataset with manageable size, e.g. 5% [df.sample(frac=0.05)]
B. Create corpus for each dataset [custom-made create_corpus() function].

3. Model Development

- Model Training
  - *[Describe the training phrase, which may include what models did you select, how you split training/validation/test sets, training epochs, and any other parameters.]*

    - The three classification models used were Logistic Regression, Naive Bayes, and Random Forest, all imported from the Sklearn library.
    - SVM was excluded as it was too time consuming and produced undesirable results.
    - Three pairs of feature matrices (train and test) were obtained via the following linguistic feature extraction algorithms: Bag-of-Words, TF * IDF, and Word2Vec.
    - All three classification models were used for each feature extraction algorithm.
    - Before fitting to the models, the data was rebalanced using imblearn's SMOTE and scaled using sklearn's StandardScaler.
    - When training the models, only the 1000 most relevant features were taken into account.
    - All models were run with standard configurations and parameters.
    - Optionally, the data can be split using the train_test_split function with a test_size parameter of 0.20. This randomizes the training data and reserves 20% of it for the test set, which results in a larger testing data set. I found this configuration to be optimal, any lower results in worse performance and any higher leads to overfitting.

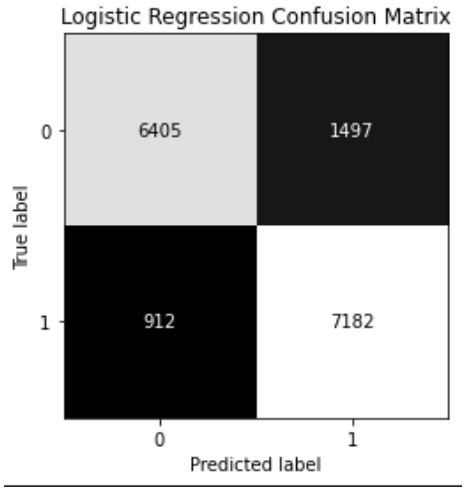
- Model Evaluation
  - *[Present the results of your models, basic evaluation looks at metrics such as accuracy, precision, or F1 score to determine which model is the best fit to solve the problem. You may find the model performance not good enough, in which case you can experiment further to improve the model performance with different features or more complex deep learning models.]*

    - After running several trials, the best performing models were typically Linear Regression and Random Forest. These generally finished with accuracy scores and AUC values near or equal to 0.9. The better performing feature extraction algorithms were–on average–TF * IDF and Bag-of-Words, though the best of Word2Vec performs just as well. The models do better when using train_test_split() to generate the testing dataset. The provided testing set may be too small. Still, it does an adequate job in classifying sentiment against the given dataset, with accuracy scores averaging 0.75. The specific results are as follows:

Bag-of-Words

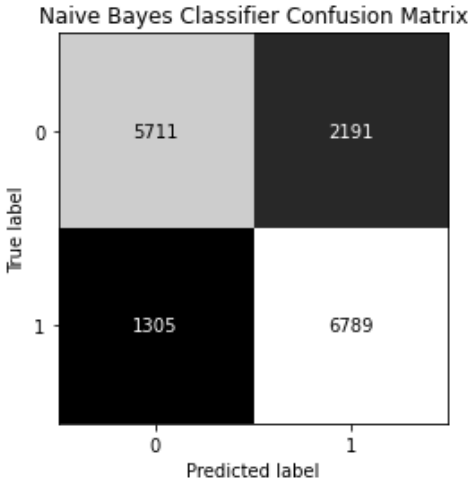
Logistic Regression

	precision	recall	f1-score	support
0	0.88	0.81	0.84	7902
1	0.83	0.89	0.86	8094
accuracy			0.85	15996
macro avg	0.85	0.85	0.85	15996
weighted avg	0.85	0.85	0.85	15996



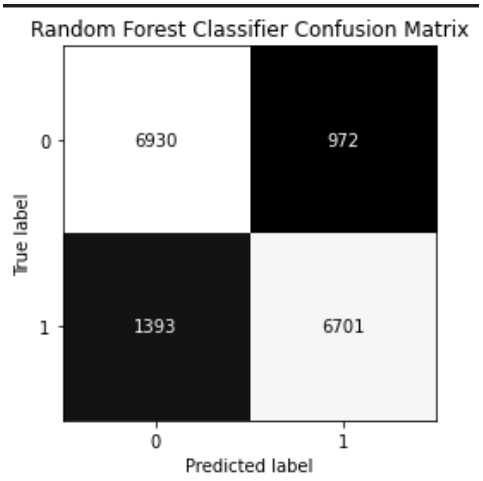
Naive Bayes Classifier

	precision	recall	f1-score	support
0	0.81	0.72	0.77	7902
1	0.76	0.84	0.80	8094
accuracy			0.78	15996
macro avg	0.79	0.78	0.78	15996
weighted avg	0.78	0.78	0.78	15996



Random Forest Classifier

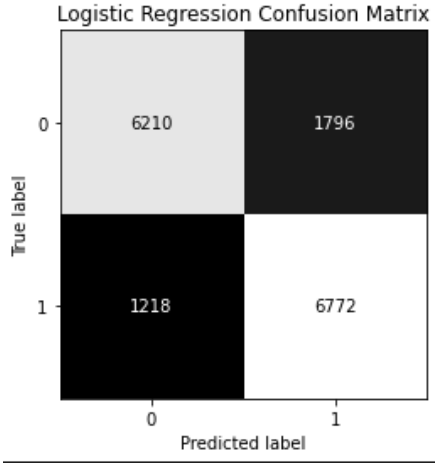
	precision	recall	f1-score	support
0	0.83	0.88	0.85	7902
1	0.87	0.83	0.85	8094
accuracy			0.85	15996
macro avg	0.85	0.85	0.85	15996
weighted avg	0.85	0.85	0.85	15996



TF * IDF:

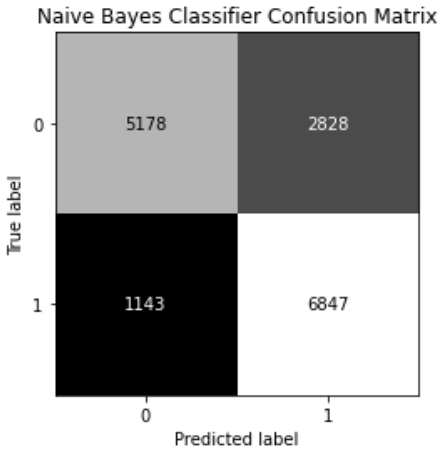
Logistic Regression

	precision	recall	f1-score	support
0	0.84	0.78	0.80	8006
1	0.79	0.85	0.82	7990
accuracy			0.81	15996
macro avg	0.81	0.81	0.81	15996
weighted avg	0.81	0.81	0.81	15996



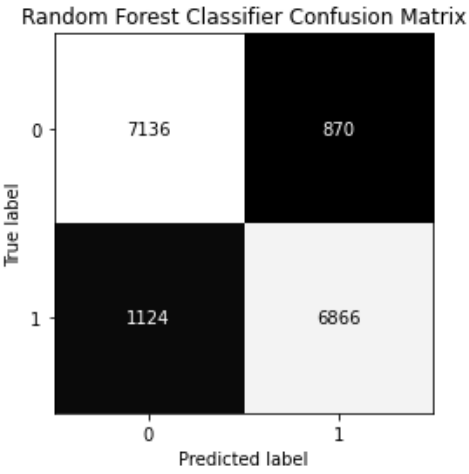
Naive Bayes Classifier

	precision	recall	f1-score	support
0	0.82	0.65	0.72	8006
1	0.71	0.86	0.78	7990
accuracy			0.75	15996
macro avg	0.76	0.75	0.75	15996
weighted avg	0.76	0.75	0.75	15996



Random Forest Classifier

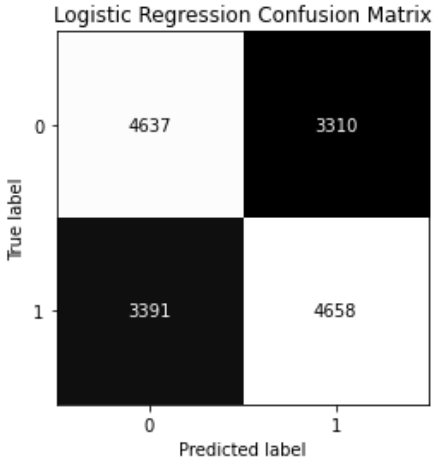
	precision	recall	f1-score	support
0	0.86	0.89	0.88	8006
1	0.89	0.86	0.87	7990
accuracy			0.88	15996
macro avg	0.88	0.88	0.88	15996
weighted avg	0.88	0.88	0.88	15996



Word2Vec:

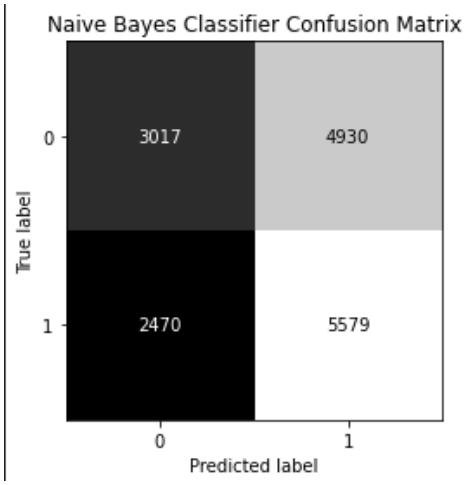
Logistic Regression

	precision	recall	f1-score	support
0	0.58	0.58	0.58	7947
1	0.58	0.58	0.58	8049
accuracy			0.58	15996
macro avg	0.58	0.58	0.58	15996
weighted avg	0.58	0.58	0.58	15996



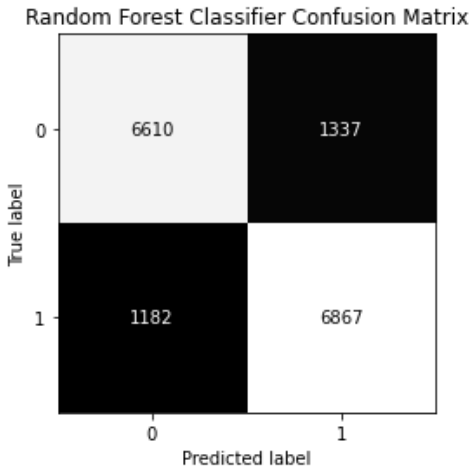
Naive Bayes Classifier

	precision	recall	f1-score	support
0	0.55	0.38	0.45	7947
1	0.53	0.69	0.60	8049
accuracy			0.54	15996
macro avg	0.54	0.54	0.53	15996
weighted avg	0.54	0.54	0.53	15996



Random Forest Classifier

	precision	recall	f1-score	support
0	0.85	0.83	0.84	7947
1	0.84	0.85	0.85	8049
accuracy			0.84	15996
macro avg	0.84	0.84	0.84	15996
weighted avg	0.84	0.84	0.84	15996



4. Discussion

- *[Does your model perform well enough? Discuss the potential reasons that your model fixes the problem well or not]*

  I believe the models performed fairly well. Some combinations are better than others. For instance, Bag-of-Words + Logistic Regression, TF * IDF + Random Forest, and Word2Vec + Random Forest, are extremely high-performing with accuracy scores of 0.85, 0.88, and 0.84 respectively. Meanwhile, the very same Word2Vec, when coupled with any other classifier, is terribly inaccurate, averaging 0.54 accuracy. With better computational power, allowing me to work with more than 5% of the data at once, I believe the models would have performed even better. Despite these limitations, the models still outperform random guessing and can predict text sentiment fairly accurately.

- *[What are the challenges you met during both the data preparation and model development processes? How did you solve them?]*

  Most of the challenges I encountered were mainly due to poor conceptual understanding and a lack of technical know-how. I definitely spent a lot of time researching how to do simple things with the libraries and an even greater amount of time trying to understand what was truly going on and how everything fit together. Another challenge was the size of the dataset. It was very time consuming to do everything, and if not careful the program could crash at any moment due to a MemoryError.

- *[Any reflections or thoughts on this assignment?]*

This was a great assignment. Highly applicable, very relevant, and just the right amount of challenging.

5. Appendix

- https://github.com/skinnii/multimodal-ml