Project Phase 2b, Project Competition Participation
Ryan Smith
11/21/24

(a) Task

Name and short description of what the task is.

Sentiment Analysis Electronic Product Review Challenge, by chawki

In this competition, participants will develop models to classify customer reviews into sentiment categories: Positive, Neutral, or Negative. This challenge is ideal for anyone interested in natural language processing and machine learning.

(b) Approaches

Describe how you approached the task. Include any preprocessing steps you needed to complete to prepare the data, how you evaluated your approaches using your own, which kinds of models you tried, how you selected hyperparameters, and so on. This should be the longest section and you may include any results/discussion from your various model runs here to demonstrate the effort that you put forth to arrive at a high-quality model for the task.

The General Approach

The general approach is of picking two main models, then hyperparameter tuning them. There won't be a real goal of the number of tunings, but it will be at least probably five tuning attempts per model. I will have the standard metrics printout in the python notebook code and later in this document. I will at the end pick the best model and tuning and print that out to the preds.txt file for submission to the competition.

Preprocessing

Basic Vectorization of the text was all that was needed, and utilized on all models.

How the Approach was Evaluated Based on My Own

I looked at this similar to how I looked at my project. The workflow was essentially:

- Dataset Loading
- Preprocessing (basic preprocessing)

- Feature Engineering (none needed with this)
- Model 1 Building
- Model 1 Training
- Model 1 Validation
    - Metrics
    - Hyperparameter Tuning
    - Metrics on the Tuned Model
    - Comparison of the Models
- Model 1 Testing
    - Metrics on the Testing Model
- Conclusion on Model 1

Then repeat this process for a second model, and a conclusion on the models and hyperparameter tunings.

This seems to be a fairly standard workflow that makes sense and makes a project go smoother.

The Models

The two best models for this are Logistic Regression (multinomial because it is not binary, there are more than two classes, but we can still use Logistic Regression) and Random Forest.

Multinomial logistic regression is an extension of logistic regression that allows for classification into more than two categories. This makes it well-suited for text classification tasks where documents need to be assigned to multiple topics, genres, or sentiment labels (e.g., positive, negative, neutral). It works by estimating the probability of a text belonging to each class using a softmax function, which generalizes the logistic function for multiple categories. The model learns a set of coefficients for each class, representing the importance of different features in predicting that class. These coefficients are combined with the feature values of the input text to produce a score for each class, and the text is assigned to the class with the highest probability. Multinomial logistic regression is favored for its simplicity, interpretability, and efficiency in handling multi-class text classification problems.

Random Forest is an ensemble learning method that leverages the wisdom of multiple decision trees to classify text. Imagine having a group of experts, each specializing in different aspects of the text, like vocabulary, grammar, or topic. Each expert (decision tree) makes a prediction based on their knowledge, and the final classification is determined by a majority vote or averaging their predictions. This approach often leads to robust and accurate results because it reduces the risk of overfitting to the training data. Random Forest is particularly useful when dealing with complex datasets and can handle large amounts of text data efficiently. Its ability to capture non-linear relationships and interactions between words makes it a valuable tool for tasks like spam detection, news categorization, and author identification.

Model 1: Logistic Regression (multinomial)

Logistic Regression

Hyperparameters

```
C = 1.0
max_iter = 1000
solver = 'lbfgs'
```

These are default values for the three hyperparameters.

C: Controls the regularization strength, where lower values increase regularization and help prevent overfitting by penalizing complex models.

max_iter: Sets the maximum number of iterations the algorithm will run to find the optimal solution, potentially stopping early if convergence isn't reached.

solver: Specifies the algorithm used to find the best model parameters, with options like 'lbfgs', 'sag', or 'newton-cg' offering different performance trade-offs for various datasets.

Tunings

We will try various tunings to get good results.

Default Values

This is with default values.

```
Validation Accuracy: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         4
     Neutral       0.50      0.25      0.33         8
    Positive       0.71      0.92      0.80        24

    accuracy                           0.67        36
   macro avg       0.40      0.39      0.38        36
weighted avg       0.58      0.67      0.61        36
```

Tuning

```
# Hyperparameters
```

```
C = 0.01
max_iter = 3000
solver = 'saga'


Validation Accuracy: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         4
     Neutral       0.00      0.00      0.00         8
    Positive       0.67      1.00      0.80        24

    accuracy                           0.67        36
   macro avg       0.22      0.33      0.27        36
weighted avg       0.44      0.67      0.53        36
```

These are giving very similar values, and a little worse actually.

Tuning

```
# Hyperparameters
C = 1.0
max_iter = 10000
solver = 'saga'
penalty = 'l1'


Validating the model...
Validation Accuracy: 0.6111111111111112
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         4
     Neutral       0.25      0.12      0.17         8
    Positive       0.68      0.88      0.76        24

    accuracy                           0.61        36
   macro avg       0.31      0.33      0.31        36
weighted avg       0.51      0.61      0.55        36
```

We added a new hyperparameter, penalty, which gave us decent results, but not as good as the default values. I have tried many changes to these four hyperparameters also, and default is still winning, interestingly.

Model 2: Random Forest

Random Forest is second.

Hyperparameters

These are default values:

n_estimators: 100
max_depth: None
random_state: None

n_estimators: Determines the number of decision trees in the random forest, where more trees generally improve performance but increase computational cost.

max_depth: Controls the maximum depth of each decision tree, limiting the number of levels and preventing overfitting by simplifying individual trees.

random_state: Sets a seed for the random number generator, ensuring reproducibility of results by controlling the randomness in building the forest.

Tunings

We will try various tunings on this Random Forest model.

Default Values

These are similar results to one of our tuned settings from Logistic Regression.

```
Validation Accuracy: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         4
     Neutral       0.00      0.00      0.00         8
    Positive       0.67      1.00      0.80        24

    accuracy                           0.67        36
   macro avg       0.22      0.33      0.27        36
weighted avg       0.44      0.67      0.53        36
```

Tuning

```
# Hyperparameters for RandomForestClassifier
n_estimators = 20000
max_depth = 5
random_state = 42
min_samples_split = 100
min_samples_leaf = 50


Validation Accuracy: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         4
     Neutral       0.00      0.00      0.00         8
    Positive       0.67      1.00      0.80        24

    accuracy                           0.67        36
   macro avg       0.22      0.33      0.27        36
weighted avg       0.44      0.67      0.53        36
```

I have changed all of these values to many different combinations, and all the code and output look ok, but they are not changing for Random Forest.

Summary

We ran various combinations on Logistic Regression Multinomial and Random Forest. Logistic Regression has the best result.

(c) Best Results

Which approach worked best? Why do you think that is?

This is Logistic Regression with default values. I tried many other types of values and hyperparameters, all were lower results, interestingly.

```
Validation Accuracy: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00         4
     Neutral       0.50      0.25      0.33         8
    Positive       0.71      0.92      0.80        24

    accuracy                           0.67        36
```

macro avg	0.40	0.39	0.38	36
weighted avg	0.58	0.67	0.61	36