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

(a) Task

Name and short description of what the task way.

HIPA-AI Competition, by a team led by Zoiya

The task is to download a Reddit posts dataset and train a model to predict whether they're HIPAA violations.

(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

I did some basic checks for missing values. The datasets were already preprocessed and ready for training with this project competition.

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 and Hyperparameters

The models I chose were Logistic Regression and SVM. Both of these models were tested with basic hyperparameter values and tuned, along with some other preprocessing tuning.

Model 1: Logistic Regression

Logistic regression is a popular and effective model for text classification tasks. It's a simple, yet powerful algorithm that's particularly well-suited for binary classification problems, where the goal is to categorize text into one of two classes (e.g., spam or not spam, positive or negative sentiment). Despite its name, logistic regression is a classification algorithm, not a regression one. It works by estimating the probability of a text belonging to a specific class using a logistic function.

Hyperparameters

Define hyperparameters
C_values
max_iter_values
solvers
penalty

C_values: Controls the penalty for complex models, balancing between fitting the training data well and avoiding overfitting.

max_iter_values: Sets the maximum number of iterations for the optimization algorithm to find the best model parameters.

solvers: Specifies the algorithm used to find the best-fitting parameters for the model.

penalties: Determines the type of regularization used to prevent overfitting by shrinking the model's weights.

Those are the three hyperparameters we are tuning. We might add more though in later tunings.

Tunings

Default Values

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

          no       0.44      1.00      0.62         4
         yes       1.00      0.17      0.29         6

    accuracy                           0.50        10
   macro avg       0.72      0.58      0.45        10
weighted avg       0.78      0.50      0.42        10
```

Tuning

```
# Define hyperparameters
C_values = [0.01, 0.1, 1.0, 10.0, 100.0]  # Different values for regularization strength
max_iter_values = [100, 200, 300]  # Different values for maximum number of iterations
solvers = ['liblinear', 'lbfgs', 'saga']  # Different solvers
```

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

          no       0.67      1.00      0.80         4
         yes       1.00      0.67      0.80         6

    accuracy                           0.80        10
   macro avg       0.83      0.83      0.80        10
weighted avg       0.87      0.80      0.80        10
```

Between default values and the ones above, we go from a .5 accuracy to a .8. Also, our other scores are better, such as macro avg and weighted avg, along with precision, recall, and f1-score.

Tuning, adding hyperparameter 'penalty'

```
Training model with C=100.0, max_iter=100, solver=saga, penalty=l1...
Validation Accuracy: 0.8
Validation Classification Report:
              precision    recall  f1-score   support

          no       0.67      1.00      0.80         4
         yes       1.00      0.67      0.80         6

    accuracy                           0.80        10
   macro avg       0.83      0.83      0.80        10
...
weighted avg       0.87      0.80      0.80        10
```

We added 'penalty' here and are receiving basically the same scores as our previous tuning. This is somewhat due to the 10 datapoints of the test set making it challenging for tuning.

Tuning, very different settings

```
# Define hyperparameters
C_values = [0.001, 0.005, 0.01]  # Much smaller values for regularization strength
max_iter_values = [50, 75, 100]  # Smaller values for maximum number of iterations
solvers = ['newton-cg', 'sag']  # Different solvers
penalties = ['l2', 'none']  # Different penalties, including no regularization
```

These drastic changes produced poor results

```
Validation Accuracy: 0.4
Validation Classification Report:
              precision    recall  f1-score   support

          no       0.40      1.00      0.57         4
         yes       0.00      0.00      0.00         6

    accuracy                           0.40        10
   macro avg       0.20      0.50      0.29        10
weighted avg       0.16      0.40      0.23        10
```

I have tested other settings, which might be too much code to put here for every hyperparameter tuning, and the previous one with an accuracy of .8 is the best I could get. This current one is the most drastic combination I tried which resulted in very poor scores.

Model 2: SVM

Support Vector Machines (SVMs) are powerful algorithms used in supervised learning for text classification tasks. They work by finding an optimal hyperplane that maximally separates different classes in a high-dimensional space. In the context of text classification, each document is represented as a point in this space, and the SVM aims to find the hyperplane that best separates documents belonging to different categories. The "support vectors" are the data points closest to the hyperplane and have the most influence on its position. SVMs are effective for text classification because they can handle high-dimensional data and are good at capturing complex relationships between words and classes.

Hyperparameters

C_values
kernels
gammas

C_values: Controls the penalty for misclassifications, balancing between maximizing the margin and minimizing training errors.

kernels: Specifies the type of function used to map data points to a higher-dimensional space where they might be linearly separable.

gammas: Determines the influence of a single training example, affecting the decision boundary's curvature and model complexity.

Tunings

About 15 total tunings, from manual ones to the pseudo grid search code that iterates through various combinations which produced that actual best result.

Default Values

These produced decent values:

```
Validation Accuracy: 0.5
Validation Classification Report:
              precision    recall  f1-score   support
```

```
              no       0.44      1.00      0.62         4
             yes       1.00      0.17      0.29         6

        accuracy                           0.50        10
       macro avg       0.72      0.58      0.45        10
    weighted avg       0.78      0.50      0.42        10
```

Those are considerably better than our tuned values, interestingly. The tuned ones are below this.

Tuning,

Here is our main tuning. I went through probably 15 or so different settings, but they produced very similar results that were poorer. I won't list all of the results here, but here is the main settings that produced the best results:

```
Best hyperparameters: {'C': 1.0, 'kernel': 'linear', 'gamma':
'auto'}
Best validation accuracy: 0.8
Validation Accuracy: 0.8
Validation Classification Report:
              precision    recall  f1-score   support

           no       0.67      1.00      0.80         4
          yes       1.00      0.67      0.80         6

    accuracy                           0.80        10
   macro avg       0.83      0.83      0.80        10
weighted avg       0.87      0.80      0.80        10
```

Summary

We tested many settings for our first model Logistic Regression, along with our second model, SVM, and each model produced our top score model results. The 10 datapoints make a large variation in results challenging, so it is not surprising we have a top result that is the same in both models. We technically will submit the one from the SVM model.

(c) Best Results

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

This tuning with our pseudo grid search produced good values. This ties our highest values from both models.

```
Best hyperparameters: {'C': 1.0, 'kernel': 'linear', 'gamma':
'auto'}
Best validation accuracy: 0.8
Validation Accuracy: 0.8
Validation Classification Report:
              precision    recall  f1-score   support

          no       0.67      1.00      0.80         4
         yes       1.00      0.67      0.80         6

    accuracy                           0.80        10
   macro avg       0.83      0.83      0.80        10
weighted avg       0.87      0.80      0.80        10
```

We have this as our highest score in the logistic regression model, so we could go with this one, or that one for our submission to the competition.

Our pseudo grid search code labeled in the comments is just going through the various combinations of possibilities with those hyperparameters. I believe, for this dataset, this score is probably as good as it will get.

Overall, the combination of balanced regularization, a simple and effective linear kernel, appropriate scaling of gamma, and effective text preprocessing with TF-IDF vectorization contributed to the high performance of the SVM model on this dataset. These settings provided a robust and generalizable model that performed well on the validation set, making it a strong candidate for submission to the competition.