I Couldn’t Find a Therapist so I tried to Build One - Part II.

Capstone 2 Final Report

Kevin Chung
05/03/2021
Springboard

Table of Contents

Problem Statement & Project Proposal

Introduction

In my previous Capstone project, I identified three core limitations of my study. The following are those three limitations:

- Imbalanced classes & lack of data
- Lack of computation power to try out other hyperparameter settings
- Absence of comparative metrics -- currently only using BERT

For this portion of the Capstone, I will attempt to overcome these limitations in three ways:

1. Web scrape data from mental health support forums for all six classes to obtain more samples.
2. Move the computing environment to a Surface Studio equipped with a cuda GPU unit.
3. Compare results from 8 different machine learning models, and apply dimension reduction to visualize and investigate the embedding space.

After these steps, I fine tune another BERT model. I will compare all of the prediction results to determine which model performs the most reliably, and also determine which of the models is the most scalable for continued training and deployment.

Scope

This paper will compare several non-deep learning models and determine their efficacy as a multiclass text classifier. Then, I will fine tune a BERT model to compare its performance.

Milestones

I have clearly defined in the above the limitations I must overcome to make this study more robust. The first and foremost milestone is to find and successfully scrape text data from mental health support forums, and preprocess it for exploratory analysis.

Once I have successfully obtained the data, I explore it in the same manner as I have previously in Capstone I, then I utilize slightly more advanced techniques to explore the feature space, using TruncatedSVD and UMAP-learn. The third milestone of this portion of the project is the comparison of the different classifiers, and to determine the best non-deep learning classifier for the datasets I introduce. Then, I retry using BERT while fine tuning to compare.

Metrics

I will leverage Scikit-Learn's classification report to obtain a high-level view of the results for each iteration. Since this is a multiclass classification task, observing the f1 score will be particularly helpful. The other metrics provided by the report will also serve as a good measure for selecting the classifier to tune. Then, I will employ a more visual approach by plotting the ROC AUC curve, as well as the confusion matrix to identify signs of performance issues and overfitting.

Data Wrangling & EDA Report

Sourcing Data

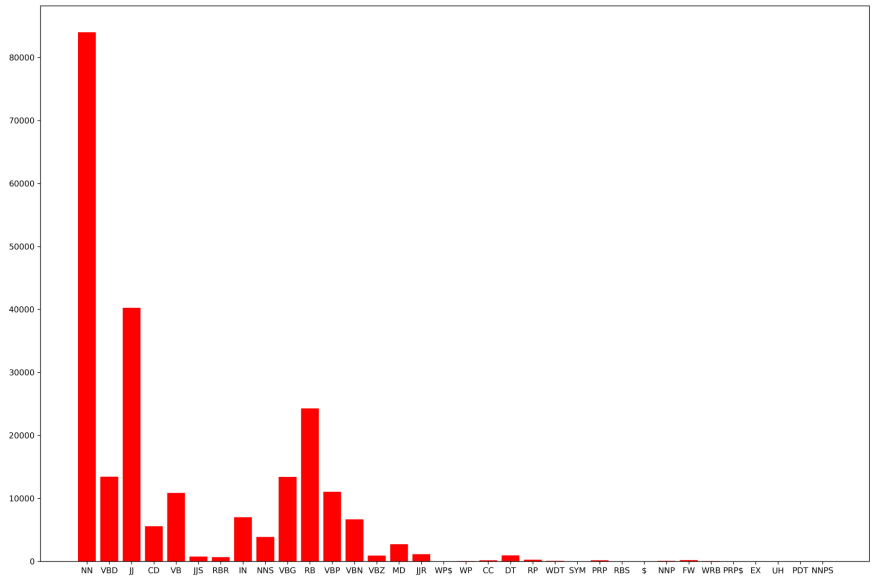
I identified a large mental health support forum called “mentalhealthforum”, which had groups for a variety of mental health conditions. The site was relatively easy to navigate, and it had groups for all six of my classes which I could not find elsewhere. A quick eye scan over the content seemed to be very rich in content; however, the number of available documents differed from one another significantly.

I used BeautifulSoup and requests to obtain the datasets. By iterating over the page count available in each of the forum groups (each group had a different number of pages available) I was able to create a nested for loop which clicked in each page and grab all of the specified text documents within that page. The biggest challenge of this phase was going through the html scripts to identify classes that indicated the correct documents on the web page.

Wrangling Data

I followed the same recipe to prepare the data for exploratory analysis. The steps involved tokenization, removing stopwords, and lemmatization. Once the data had been preprocessed, I proceeded to take a quick look into the corpus for visual insights.

I generated a count of the 20 most frequent words that showed up, searched for concordances with the word “want”, then found quadgram collocations. I also POS tagged every document to visualize choices in diction, and how they vary by class, by plotting each tags’ frequency throughout the dataset. The below are the outputs for the depression forum.

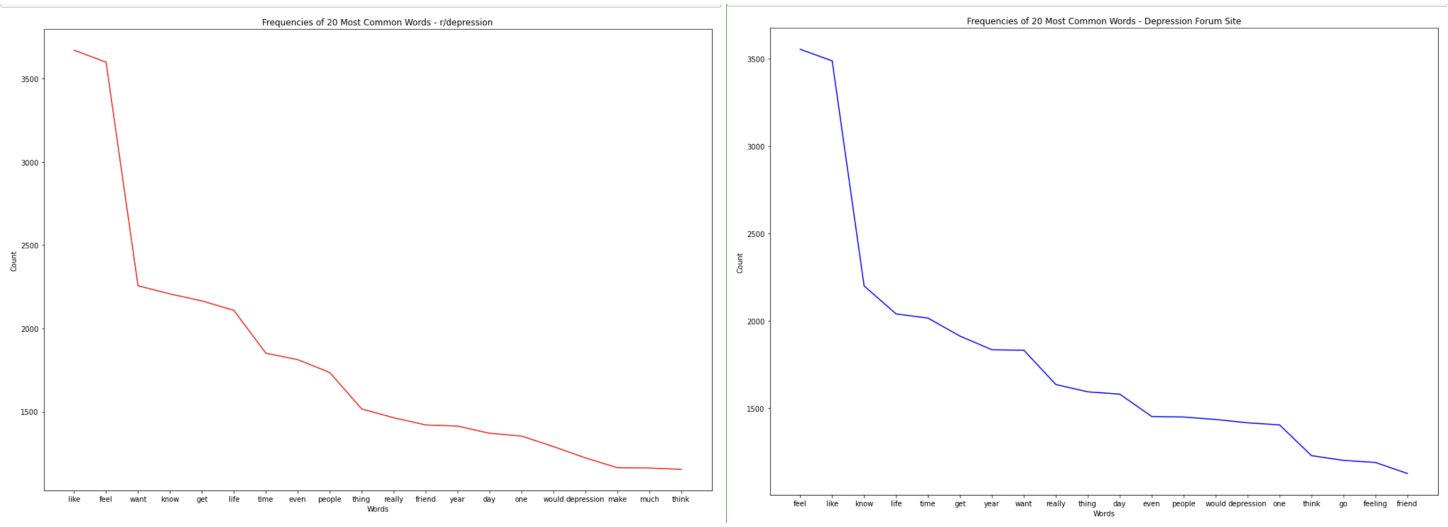


```
[('feel', 3555), ('like', 3489), ('know', 2201), ('life', 2041), ('time', 2017), ('get', 1914), ('year', 1836), ('want', 183
3), ('really', 1637), ('thing', 1595), ('day', 1582), ('even', 1454), ('people', 1451), ('would', 1437), ('depression', 141
8), ('one', 1406), ('think', 1230), ('go', 1203), ('feeling', 1191), ('friend', 1128)]
ing seems easier way much doubt ever want share experience read lot topic time
l think ruined honor locking 3 month want go back work living house live diffe
afraid tired everyday like nightmare want lose life person fell love know plea
en week really struggling experience want come home feel like need stay longer
ng promoted afford monthly rent plus want emergency saving fund live mother cl
ybody relate advice thanks nan hi dr want sign two week go ahead go work dr gi
ress let honest get lot stress would want come home hurt see depression suicid
ul dreamy 3 hour ever rare feel like want want wake much better time asleep sa
eamy 3 hour ever rare feel like want want wake much better time asleep sad dep
e name know others similar situation want tell last spend sunny day loved one
losing mom one trust world feel like want die hopeless starting get anger issu
onfident sexy hate negative attitude want stop therapy support group gonna get
d air forcing pretend happy positive want throw hand say amazing instead looke
ed got mad said really uncomfortable want talk immature attitude therapist alw
esteem hard hidradenitis suppuritiva want negative hi taken day today really s
ontact therapist reach fucking sleep want die though want hurt lovely boyfrien
 reach fucking sleep want die though want hurt lovely boyfriend really want di
gh want hurt lovely boyfriend really want die live like anymore either please
k long time figure never wanting man want woman suck social expectation abuse
 wanting easy time think lot anymore want die intend anything speed process wa
<nltk.collocations.QuadgramCollocationFinder object at 0x000001C26DF9EC48>
[(('faint', 'faint', 'faint', 'faint'), 11), (('anyone', 'else', 'feel', 'like'), 9), (('make', 'feel', 'even', 'worse'),
6), (('diagnosed', 'major', 'depressive', 'disorder'), 5), (('cut', 'long', 'story', 'short'), 5), (('nothing', 'make', 'hap
py', 'anymore'), 4), (('want', 'get', 'better', 'want'), 4), (('every', 'day', 'feel', 'like'), 4), (('feel', 'like', 'stayi
ng', 'bed'), 4), (('feel', 'like', 'good', 'enough'), 4), (('want', 'go', 'back', 'work'), 3), (('wish', 'could', 'go', 'bac
k'), 3), (('also', 'feel', 'like', 'im'), 3), (('thank', 'taking', 'time', 'read'), 3), (('say', 'something', 'along', 'lin
e'), 3), (('really', 'want', 'go', 'back'), 3), (('feel', 'like', 'want', 'exist'), 3), (('many', 'people', 'around', 'worl
d'), 3), (('1', '5', 'year', 'ago'), 3), (('time', 'playing', 'video', 'game'), 3)]
```

One interesting discovery is the similarity between the reddit data and the forum site's data. When I compared word clouds, the outputs were nearly identical!

A side-by-side comparison of word clouds from two depression data sources.

Down to word choice and term frequency ratios, the corpuses' resemblance to each other are uncanny even to the naked eye. I also plotted and compared the top 20 word frequencies from both the subreddits and the forum sites. The below were the results:



The left image is the word frequency from r/depression subreddit, and the right is from the forum site. I repeated this comparison for each of the classes and found similar results.

I then combined the two datasets into a master dataset, called "master_rf.csv". This helped to nearly double the amount of examples available for training, thus taking care of the first of the three limitations.

Machine Learning & Dimensionality Reduction

Procedures

In this section, I will examine model comparisons on three datasets created from the previous section: **master - subreddit**, **master - forum**, and **master - combined**. Then, I explore the feature space of the combined data to understand the effects of dimensionality reduction.

Master - Subreddit

The master dataset is the final dataset with two columns, text and label. Documents present in this dataset are exclusively collected and processed from subreddits for each of the classes. I constructed a dictionary of 9 models I wanted to compare against each other with default parameter settings. The tuning will come later. The 9 models are as follows:

- MultinomialNB()
- SGDClassifier()
- LogisticRegression()
- RandomForestClassifier()
- AdaBoostClassifier()
- GaussianNB()
- DecisionTreeClassifier()
- KNeighborsClassifier()
- DummyClassifier()

By iterating through each fit and using scikit-learn's classification_report, I was able to identify SGDClassifier(), LogisticRegression(), and RandomForestClassifier() as the three best performers for this dataset, and the remaining two datasets. For brevity I only included the results from top 3 performing classifiers.

SGD - Classification Report

Classes	Precision	Recall	f1-score
0	.78	.80	.79
1	.84	.77	.80
2	.79	.74	.76
3	.88	.79	.83
4	.67	.92	.78

5	.87	.78	.82

Accuracy = 80%

LOG - Classification Report

Classes	Precision	Recall	f1-score
0	.74	.77	.76
1	.81	.73	.77
2	.74	.70	.72
3	.83	.76	.79
4	.69	.89	.78
5	.81	.76	.78

Accuracy = 77%

RF - Classification Report

Classes	Precision	Recall	f1-score
0	.69	.85	.76
1	.88	.79	.83
2	.84	.77	.80
3	.85	.83	.84
4	.77	.90	.83
5	.93	.78	.85

Accuracy = 82%

Master - Forum

SGD - Classification Report

Classes	Precision	Recall	f1-score
0	.63	.74	.68
1	.73	.58	.65
2	.74	.81	.77
3	.65	.68	.66
4	.79	.65	.71
5	.62	.57	.59

Accuracy = 68%

LOG - Classification Report

Classes	Precision	Recall	f1-score
0	.61	.74	.67
1	.80	.46	.58
2	.81	.74	.77
3	.57	.72	.64
4	.71	.67	.59
5	.75	.40	.52

Accuracy = 66%

RF - Classification Report

Classes	Precision	Recall	f1-score
0	.58	.75	.66
1	.73	.52	.61
2	.77	.68	.72
3	.55	.65	.59
4	.73	.69	.71
5	.64	.24	.35

Accuracy = 64%

Master - Combined

SGD - Classification Report

Classes	Precision	Recall	f1-score
0	.72	.74	.73
1	.76	.71	.73
2	.72	.72	.72
3	.71	.74	.73
4	.76	.79	.77
5	.79	.72	.75

Accuracy = 74%

LOG - Classification Report

Classes	Precision	Recall	f1-score
0	.68	.76	.72
1	.79	.63	.70
2	.71	.66	.69
3	.74	.71	.73

4	.65	.80	.72
5	.79	.69	.74

Accuracy = 72%

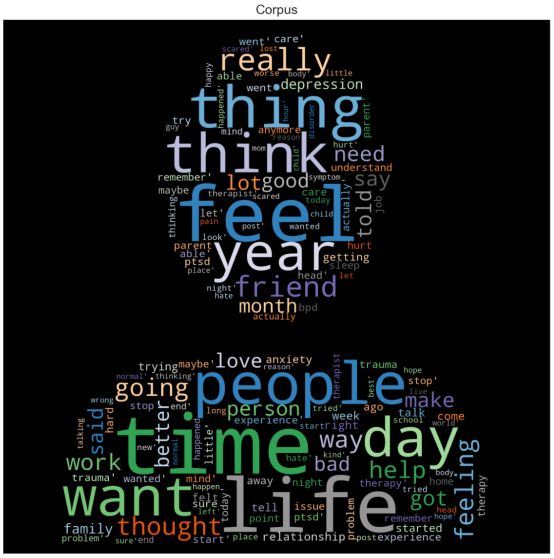
RF - Classification Report

Classes	Precision	Recall	f1-score
0	.60	.81	.69
1	.84	.69	.76
2	.80	.64	.71
3	.66	.76	.70
4	.76	.77	.76
5	.89	.61	.73

Accuracy = 72%

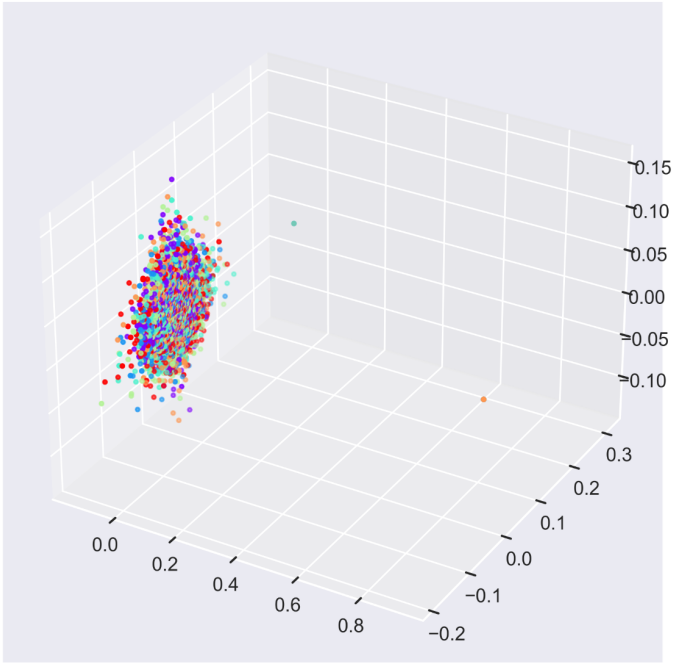
The Analysis

Apart from repeating the steps I followed in the previous section, I wanted to observe the embedding of my dataset in a more visual way. I introduce in this analysis, two dimensionality reduction techniques: TruncatedSVD, and UMAP-learn.

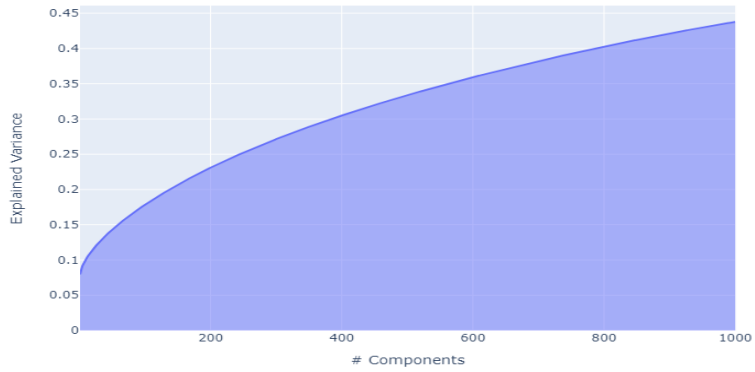


First, I Split the entire dataset before vectorizing it. I set limits on the maximum amount of documents a term can show up before it is removed from the vectorization. By doing so, I am able to reduce the dimensions from being hundreds of thousands of features to humble tens of thousands.

Because the dataset is so sparse [X_train.shape = (19005, 44744)], I expected TruncatedSVD to

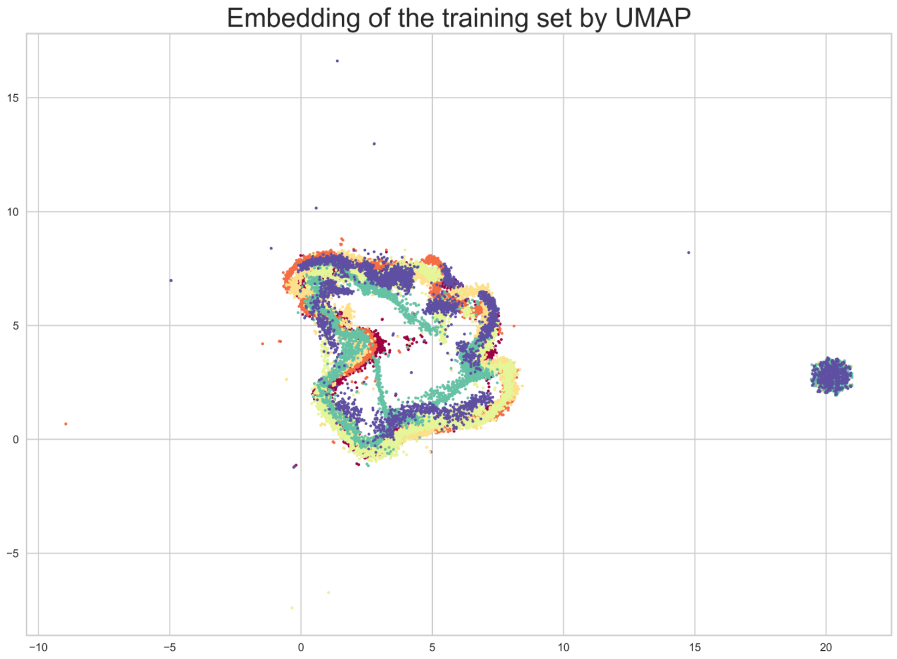


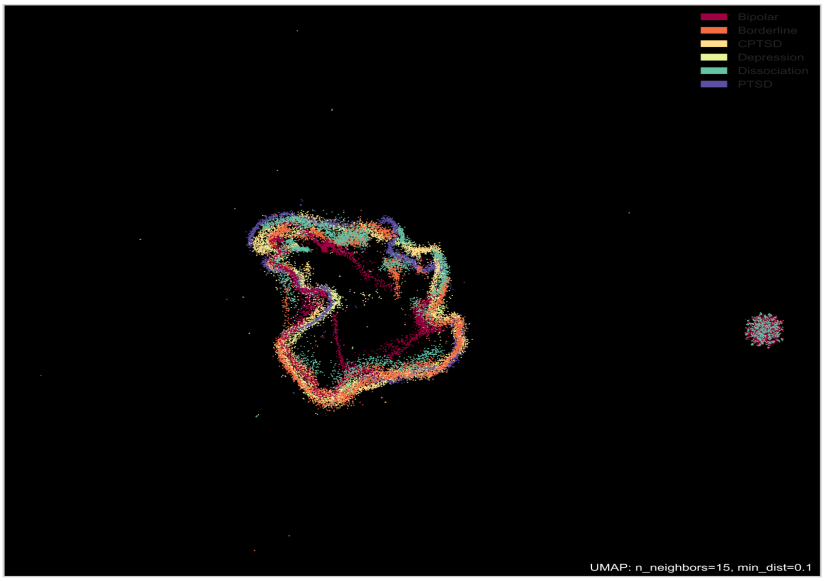
struggle, even if it is capable of crunching sparse data inputs. After I transformed the data using TruncatedSVD, I plotted a scree plot to check the explained variance. Not surprisingly, even at n_components = 1000 the explained variance was at only ~44%.



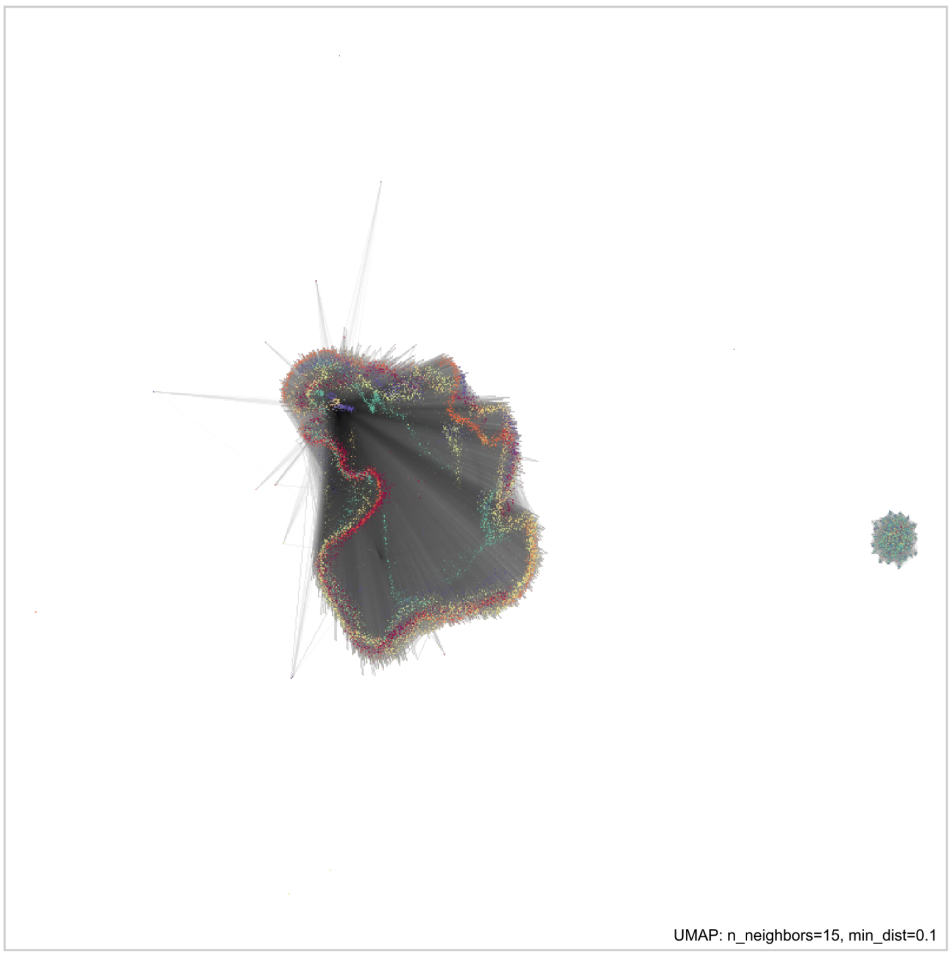
The evaluation results gathered from the model dictionary from Capstone 1 yielded only slightly better results than the dummy classifier. The best result was yielded by SGDClassifier() and LogisticRegression(), with a prediction accuracy score of merely 29%.

UMAP-learn achieved slightly better results. Projected onto two dimensions, as I did before with PCA, the data transformer achieved 56% with the KNearestClassifier(). UMAP was also helpful for understanding the embedded space, as it allows users to plot interactive plots and visualize manifolds that respects the overall topology of the data. Its flexibility in reducing dimensions also allows for dynamic distributions making it easy for the naked eye to see class separations.

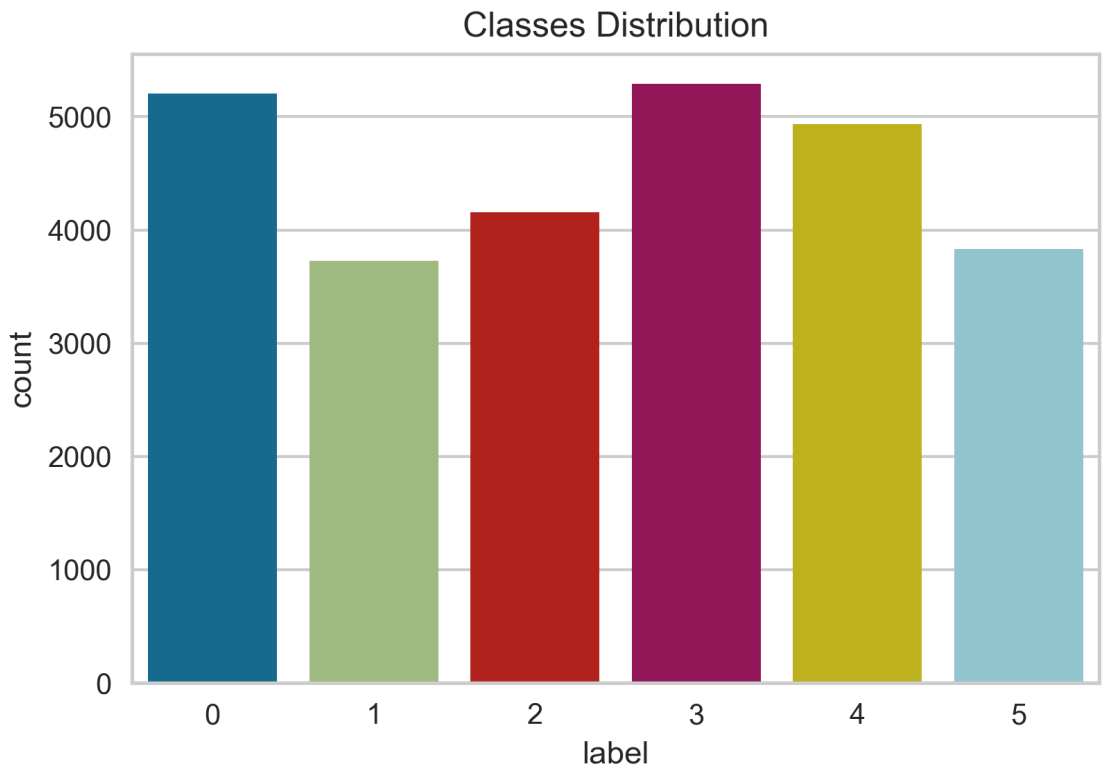




UMAP also supports visualizing connectivity plots which provides a view of the embeddings, and where it was sampled from. The graph below shows the embeddings converging to a single point, with the exception of a cluster of outliers.



Finally, I check the distribution of classes. Knowing that the imbalances in the classes affected the results in Capstone 1, I deliberately saved the step of balancing out classes for the next portion of the project which employs deep learning.



The imbalance slightly improved with the addition of the forum site data, but classes 0, 3, and 4 are still entire heads over the others.

Huggingface BERT Fine Tuning

Procedures

In this last section of the Capstone project, I use the bert-base-uncased model from the Huggingface library, and log the appropriate metrics using Weights and Biases (wandb).

First step, I make sure to split the dataset before embedding the dataset to prevent data leakage. I stratify on the y label in order to preserve the ratio of class distributions in the test data which helps with imbalances. This time, I allocate only 15% of the dataset for validation as opposed to 30% from the previous section. I want to allow BERT to have as many samples it needs to learn.

Since I am using pytorch for this portion, I had to create the proper encodings to load the dataloader objects. I used BERT's batch_encode_plus module to encode three datasets - X_train (training dataset), X_val (validation dataset), and df2 (holdout dataset set aside from when I made the **master_rf** dataset.

Once the datasets have been encoded, the next step is to convert them into tensors to load into the dataloaders. The tensors require three things: input ids, attention masks, and labels. Each are provided from the encodings from the previous steps, and they simply need to be selected and assigned to variables. Then, they are converted to tensors by using TensorDataset() and inputting the variables.

Although I'd ideally like to experiment with multiple hyperparameters, I am once again confronted by a lack of computational resources. Even though I switched the environment to a more powerful computer equipped with Nvidia GeForce GTX 1070 I kept encountering memory errors when experimenting with the batch size which meant I'd have to keep the size to a minimal. After several attempts I found out the max batch size my environment can support is 4. With such a low batch size, I am also limited in terms of the time it would take to train each epoch.

In subsequent projects I'd like to try using DistilBert, a lightweight version of BERT, or other alternatives such as RoBerta, or AlBert to circumvent the lack of resources. Unfortunately, Google Colab was not a viable option in my case. But I could attempt other cloud computing platforms in the future. After setting the batch size to 4, I created the dataloader objects using DataLoader(). Then I proceeded to define the optimizer and scheduler - both of which I had to experiment with manually.

There were two parameter grid searching modules I considered: Huggingface's Trainer(), and wandb's Sweep. Neither option worked for me due to CUDA errors which made both options default to computing via the CPU. This made my computer crash several times.

I chose Adam Weight Decay as the optimizer with a learning rate of .00005. I originally considered an even smaller learning rate which could have improved the results marginally. I defined the scheduler with the total steps equaling the length of the training set multiplied by 5 epochs, which is (21720 * 5) = 108,600.

Metrics

I predefined two methods for evaluating and tracking the performance of the model after each epoch. The two methods tracked a weighted average f1-score, and the accuracy score for each class. In my evaluation loop, I initialized wandb and logged the ROC curve which visually tracked the model's performance throughout its training phase. I also logged the validation loss to compare it to the training loss. I wanted to make sure to prevent overfitting, which had been a major issue with experimentations in previous sections of the project.

The classic measures were defined and in place. But what about the issue with imbalanced classes? F1-scores are extremely sensitive to true positive counts. Since the f1-score ignores the count of true negatives it is possible that the f1 score would inflate the performance of the classifier. Furthermore, a multiclass task working with imbalanced classes such as this one is especially susceptible to this risk. Imagine if the classifier was predicting true positives with great accuracy, but unbeknownst to us, it is actually terrible at classifying true negatives? To measure the efficacy of the model on an imbalanced dataset, it is critical that we pay attention to its classification strength of distinguishing true negatives as well. Matthew's Correlation Coefficient therefore addresses this issue by including true negatives into its calculation.

$$F1\ \text{score} = \frac{2TP}{2TP + FN + FP}$$

$$MCC = \frac{TN \times TP - FP \times FN}{\sqrt{(TN + FN)(FP + TP)(TN + FP)(FN + TP)}}$$

In addition to MCC, I use one more metric to ensure the validity of the model. Hamming Loss is the fraction of incorrectly predicted labels to all labels in the validation dataset. The lower the Hamming Loss, the better. Lastly I plot the results on a confusion matrix to gain a visual understanding of the results.

Once everything had been set, I called the train() function and waited.

Results

By the 5th epoch, the weighted f1-score was at ~78.6%, while the [validation : train loss] ratio was at [1.42 : 0.16]. I decided that lowering the train loss with more epochs would risk overfitting and capped it there.

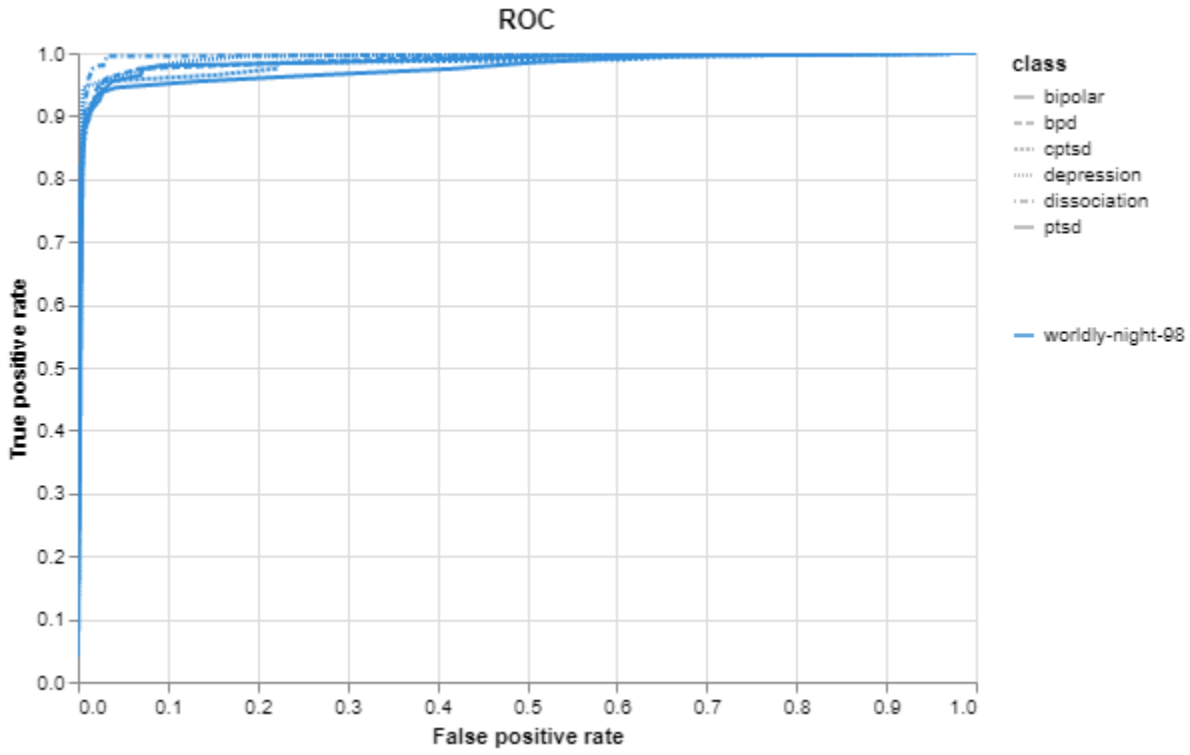
Running predictions over the holdout dataset afterward made sure to evaluate the validity of the model without data leakage. The classifier outperformed all of my expectations.

```
Class: depression
Accuracy: 95/100

Class: ptsd
Accuracy: 92/100

Class: cptsd
Accuracy: 93/100

Class: bpd
Accuracy: 91/100

Class: bipolar
Accuracy: 93/100

Class: dissociation
Accuracy: 94/100
```

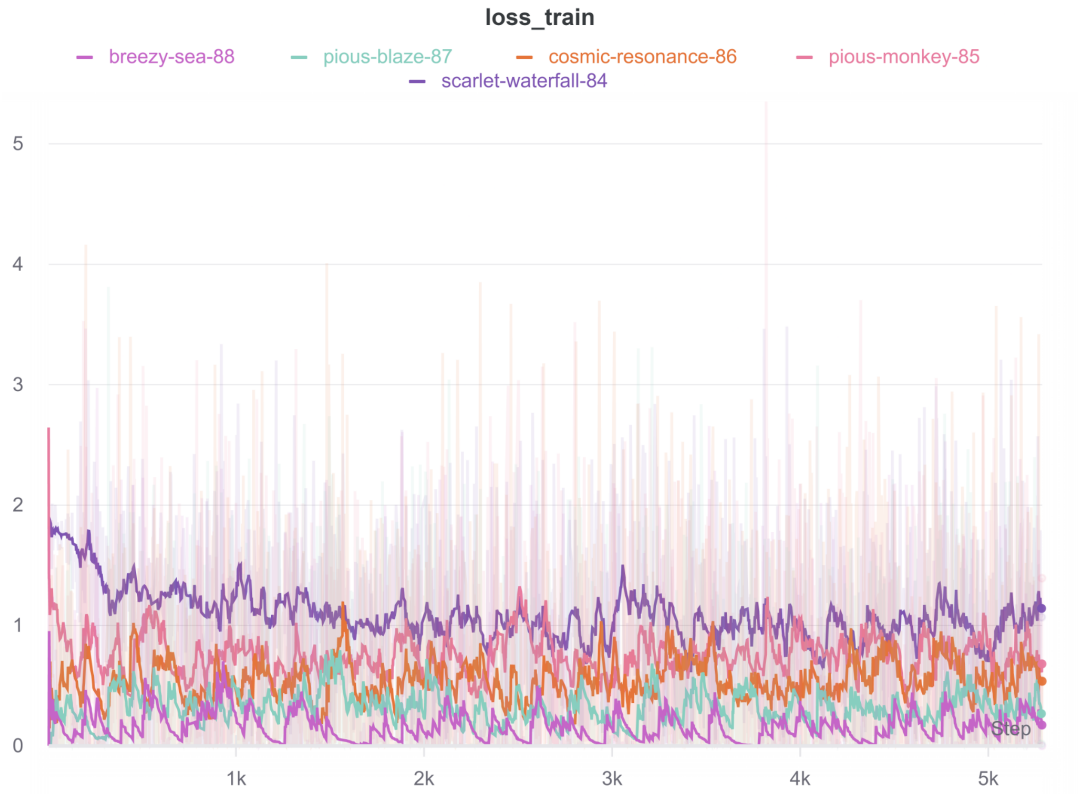
	precision	recall	f1-score	support
0	0.94	0.95	0.95	100
1	0.94	0.92	0.93	100
2	0.92	0.93	0.93	100
3	0.90	0.91	0.91	100
4	0.92	0.93	0.93	100
5	0.96	0.94	0.95	100
accuracy			0.93	600
macro avg	0.93	0.93	0.93	600
weighted avg	0.93	0.93	0.93	600

```
Matthews Correlation Coefficient: 0.9160183205496184
hamming loss: 0.07
```

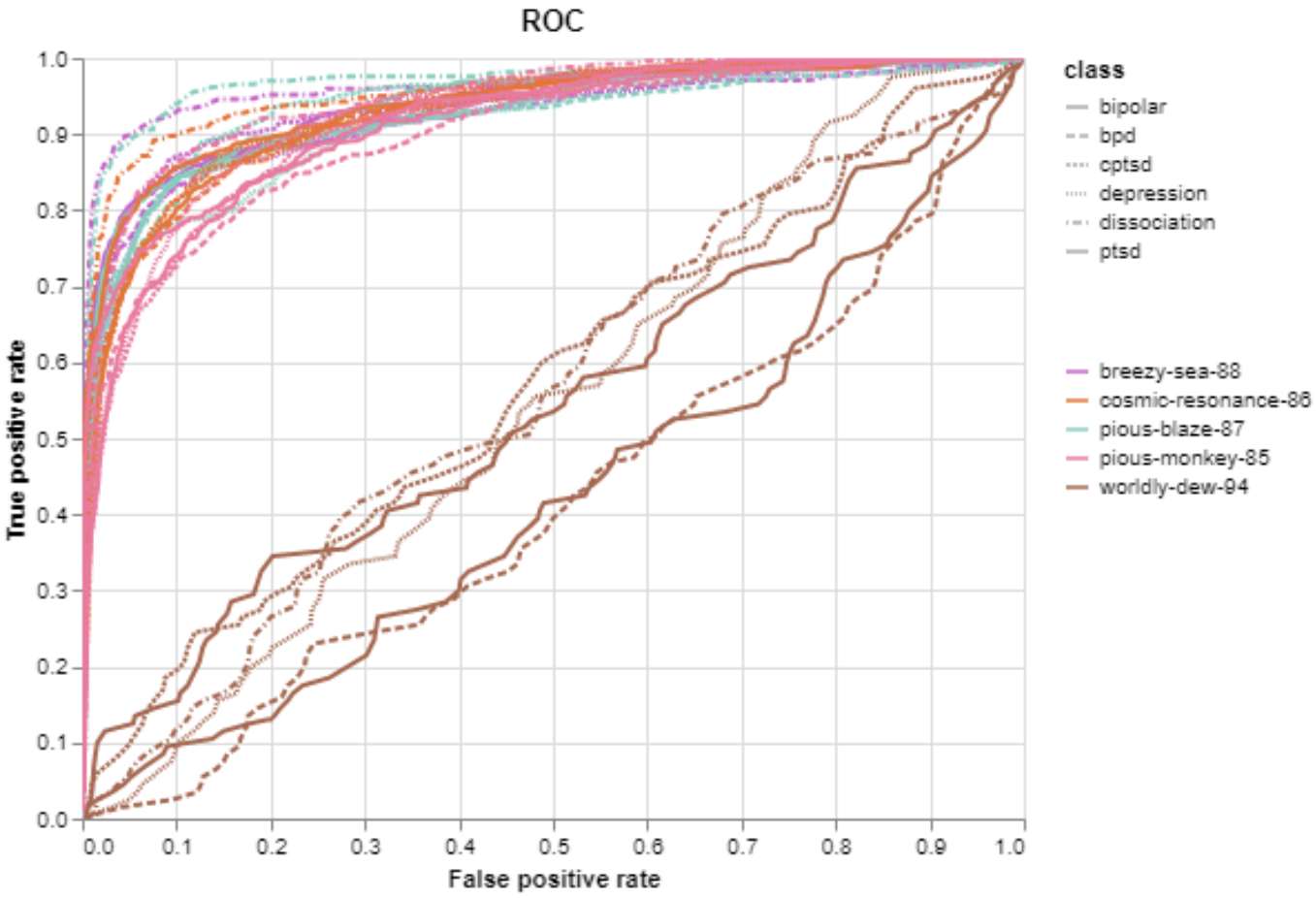
Even with the limitations of having an imbalanced dataset, the classifier achieves an impressive 93% f1-score, and 91.6% MCC. Weights & Biases helpfully charted the ROC curves for the prediction over the holdout data set:



We can also observe the decreases in training loss after each epoch in this graphic:



For fun, I also charted the validation ROC curves after each epoch against a baseline where I applied bert-base-uncased to predict classes in my dataset without training (shown with brown lines).



Discussion

In this project I demonstrated several machine learning techniques and measured their performances on a multiclass text classification task. Classifying mental health diagnoses is a challenging task even for professionally trained humans, but the output scores continuously exceeded my expectations.

As the developer, I found that non-deep learning models show much potential for simple and quick classifications. Although their results may not be as robust as deep learning methods, they are quicker to deploy and tuning may increase their effectiveness.

I found that reducing dimensionality in this case worsened the overall performance of the classifiers. This is to be expected due to much of the features being lost by being projected to a lower dimension.

Finally, I fine tuned the BERT model for a final comparison. The results were stunning, and it boggles my mind to know the result has room to improve by parameter grid searching - something I unfortunately could not do. In the future I plan on repeating the experiment with DistilBert which is a lightweight version of BERT which will allow me to grid search. This will enable me to scale this project to add more classes to the dataset.