Sentiment Analysis for Youth Mental Welfare

Vincent Qin

ABSTRACT

Mental health concerns among youth are becoming increasingly prevalent, with 20% of United States adolescents experiencing mental health problems[1]. A potential indicator of mental health concerns includes when a person's texts express overwhelming sadness or hopelessness. I present a comparison of methods to determine the emotional polarity of text. The models are trained on the Stanford SST2[2] and IMDb[3] datasets as they are based on movie reviews, which exhibit particularly apparent emotions. The data is then encoded using a Bag-of-Words (BoW) strategy by only encoding the 10,000 most common words. I tested five models: a decision tree, a random forest, the Adaboost classifier created with scikit-learn[4], a feedforward neural network with two hidden layers created using the PyTorch module[5], and a fine-tuned version of the model DistilBERT[6]. The results are cross-validated by dividing the data into ten shards, training the model on nine shards, testing it on one shard, and then repeating this procedure for every shard. Finally, the models' accuracies were compared. The DistilBERT model had the highest overall accuracy (94.89%), which made it the most suitable for large-scale classification tasks. However, the DistilBERT model has very high learning (613 m) and inference times (38 ms), which makes it inefficient for smaller tasks. Instead, I recommend the slightly weaker neural network and Adaboost models. Although they have lower accuracies (88.79% and 80.21%, respectively), their short learning time (~1-2 hours) and inference times (<10 ms) are suitable for smaller tasks that can be manually verified.

Keywords: machine learning, artificial intelligence, sentiment analysis

INTRODUCTION

In 2021, the American Academy of Pediatrics, American Academy of Child and Adolescent Psychiatry, and Children's Hospital Association declared a national emergency in child and adolescent mental health[7]. A youth's presence is heavily digital, consisting of texts, social media posts, and other interactions that can be logged. The National Institute of Mental Health finds that sadness and hopelessness are the most significant signs of depression[8]. Consequently, these signs appear in their online activities. This paper seeks the best model to draw awareness to these signs of depression.

To conduct a fair comparison, several decisions must be made. The learning method will have the most significant impact on the results. I chose to use supervised

learning as there are several free online datasets for sentiment analysis, such as the Stanford IMDb and SST2 datasets. This allows for closer observation of the model's performance as the data labels are known. I chose the aforementioned datasets for three reasons. First, the datasets are based on movie reviews which display strong emotions and are classified into 0 (negative) or 1 (positive). Second, there are around 100,000 samples, which is more than enough to train a robust model. Third, everyone's online presence is different. Some might text in short messages with slang, while others might text very formally. It is the programmer's responsibility to consider this behavior, and I did so by using two datasets. It is also beneficial that the class distribution of the datasets mimics that of real life. Next, it is crucial to decide what type of encoding to use. This paper discusses three main forms of encoding, namely: one-hot encoding, bag of words encoding, and transverse frequency-inverse document frequency encoding (TF-IDF)[9]. Each type of encoding adds information to the resulting data vector and can highly impact the model's performance. Finally, we must decide what models to use. In this paper, I explored decision trees, random forests, an implementation of the AdaBoost algorithm, feedforward neural networks, and a fine-tuned version of DistilBERT. Every model has its strengths and weaknesses. My goal is to determine which one is best at classifying emotional polarity in large sets of text messages.

CODE

The code for training the models and the datasets I created are linked on GitHub.

METHODS

Datasets

I opted to use supervised learning for my model as my model is very complex. Although sentiment analysis models are traditionally trained on unlabeled data, supervised learning offers many advantages, the most prevalent of which is extracting patterns in the model's behavior to debug and improve the model's accuracy.

Everyone's messages are unique. Some people might be very formal and use proper punctuation, grammar, and sentence length. Others might utilize slang and divide their messages into many short texts. Therefore, I opted to append the Stanford SST2 and Stanford IMDb datasets for a more balanced spread of length and formality. Since the data now had short and long sentences, I set a threshold of 100 words and appended it as a new feature. I also chose these two datasets in particular because the emotional polarity distribution of online messages is very even. A study sampling 11

billion messages from the internet[10] found that 6.51% were positive, 87.09% were neutral, and 6.40% were negative. Since the ratio of positive and negative classes is around 53:47, I concluded that the data closely mirrored the emotional distribution of polarized digital communications.

I was curious about the accuracy of the models on the data that I created. Communication on the Internet is not always formal and long-winded. I made 30 short samples and labeled them like the SST2 and IMDb Datasets. I was also curious about the models' accuracy on AI-generated data. Thus, I had ChatGPT create a separate dataset of 170 messages that were labeled similarly to the one I created.

Tokenization

It is important to divide the data into smaller characters, words, or sentences called tokens so the model can recognize patterns. This process is known as tokenization. I first tried character-based tokenization. I divided every input into its component characters using Python's .split() method. Letters, numbers, and punctuation were all divided into their own token. However, the model could not learn much from the singular characters. The model achieved an accuracy similar to random guessing due to the lack of information. Therefore, I focused on word tokenization, which is superior to character-based tokenization since each word contains a meaning. I used the Natural Language Toolkit (NLTK)[11] library's word_tokenize method to make each word, special character, number, and punctuation its token.

Encoding

Computers do not understand English. Instead, they must convert the words into numbers and perform operations on the numbers. To create these numbers, we must take a crucial step known as encoding. To start, I used one-hot encoding (OHE). In OHE, each word is represented by a binary vector. Every element is set to zero in the vector except for the index corresponding to the word. While this method is easy to implement, it has many limitations. In particular, the resulting matrices are massive, and the information is sparse when the vocabulary size is large. The model could not learn much from the data, and its accuracy was limited.

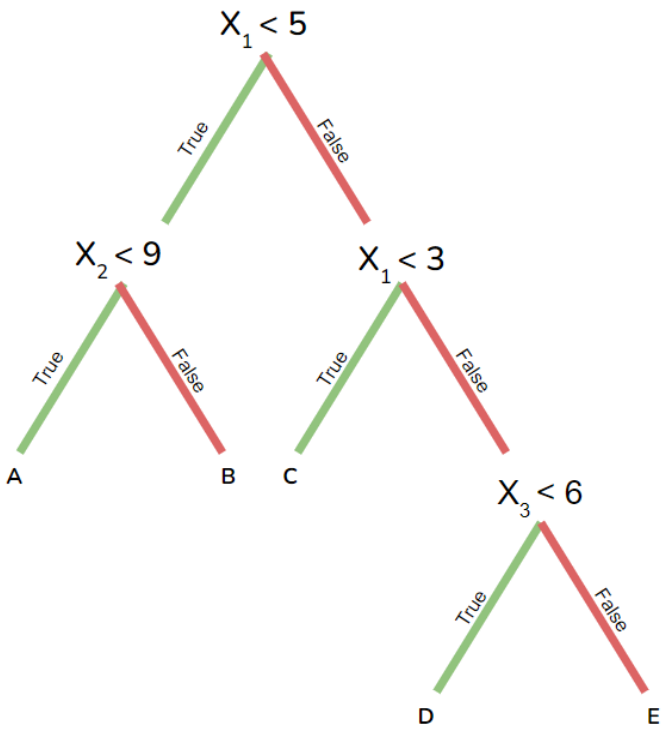
I used the Bag of Words (BoW) approach following one-hot encoding. Unlike one-hot encoding, BoW values the frequency of every word in the sample. The order of words is not saved. Each sample is represented by a vector where each element corresponds to the frequency of a word in the sample. BoW can better understand the importance of every word in the sample but can also result in large and sparse vectors if

the vocabulary is large. It also fails to consider the relative importance of words across every sample. Even so, every model performed significantly better using BoW than OHE because BoW encodes orders of magnitude more information than OHE.

In response to the shortcomings of BoW, I switched to using Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF improves BoW by considering the frequency of a word in a sample and how common that word is across all samples. This results in a value that emphasizes important words unique to a single sample while devaluing the common words across many documents. However, TF-IDF only achieved slightly better results than BoW. This is due to the complexity of the data. The large amount of words and samples makes the increase in information less significant, as the resultant vectors are still large and sparse.

Learning Algorithms

Since we are classifying the emotional polarity of a given sample, it is best to look into binary classifiers. Binary classification models work best for this task because their output is 0 or 1. They are also easy to train using the scikit-learn and PyTorch libraries and are computationally efficient.



Decision tree classifiers operate by recursively splitting the dataset into subsets while trying to maximize a criterion. The two most commonly used criteria are:

1) Information Gain[12]

Information Gain measures the decrease in entropy of a parent node and its children after a split. Splitting a dataset based on a feature shows how much information is gained. For a node x, its entropy is given by the formula

$$Entropy(x) = -\sum_{i=1}^{k} p_i log_2(p_i)$$

where p_i is the proportion of the sample in class i at node x and k is the number of classes. The information gain for a split is then

$$Information\ Gain = Entropy(parent) - \sum_{i=1}^{m} \frac{n_j}{n} Entropy(child_j)$$

where n_j is the number of samples in the child node j, n is the number of samples in the parent node, and m is the number of child nodes. Higher Information Gain indicates a better split. The split with the lowest entropy in child nodes is chosen.

2) Gini Index[13]

The Gini Index is the measure of the impurity of a node. It quantifies how often a randomly chosen sample would be incorrectly labeled if it were randomly labeled based on the class distribution of the dataset. For a node x, the Gini Index is defined as

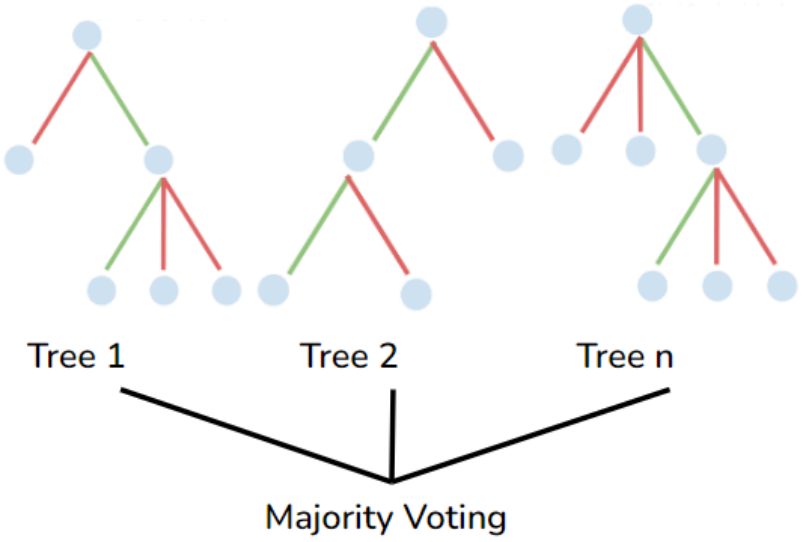
$$Gini(x) = 1 - \sum_{i=1}^{k} p_i^2$$

where p_i is the proportion of samples of class i at node x. The weighted average of the Gini Index of each split is then compared, and the split with the lowest weighted average is used.

The goal of a decision tree is to divide the data into subsets that contain the same label, which makes it comparatively simple to implement. By increasing the maximum depth, we can increase the number of splits a decision tree does and subsequently increase its complexity. Due to the complexity of the data, however, the model cannot recognize patterns before it becomes prone to overfitting.

To counter overfitting, programmers often use pruning algorithms that remove the hyper-specific splits. This is often paired with a random forest classifier, a model that

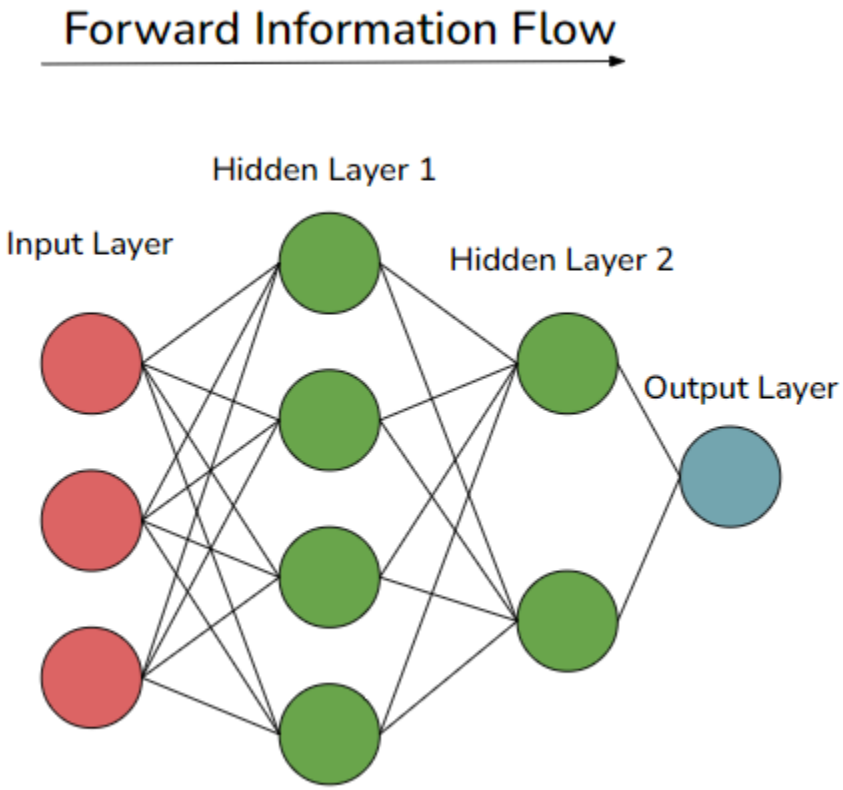
uses the aggregated vote of many decision trees. I decided not to implement a pruning algorithm because it risks underfitting the model by oversimplifying the model. Also, its impact would be minimal since its base model, a decision tree, is unable to make proper predictions at any depth. To have differently trained decision trees, the data is randomly sampled in a process known as bootstrap sampling to create multiple unique training subsets. Each subset is then fed into a decision tree. The final prediction is made by finding the majority vote across every tree. Random forests are generally more accurate and robust than singular decision trees, especially when working with large and noisy datasets.



To further improve the accuracy of the models, I used the AdaBoost algorithm through the sklearn.ensemble library. An AdaBoost classifier iteratively adds weak learners to a model to predict the difference between the predicted and target values. The model continues to fit weak learners until a stopping condition is met. Stopping criteria include a fixed number of iterations, a lack of improvement, and convergence of the loss function.

Another strong binary classifier is a neural network. I used a Feedforward Neural Network specialized for binary classification tasks. The network consists of an input layer, two hidden layers, and an output layer. The input layer receives a vector representing the sample. Each hidden layer contains neurons that apply a linear transformation to the inputs followed by the Rectified Linear Unit (ReLU) activation function. The output layer has a single neuron with a sigmoid activation function, producing the probability that the sample is classified as 1. The network is trained using

the binary cross-entropy loss function, which measures the difference between predicted and actual labels. I used the AdamW optimizer, which updates the network edge weights based on the loss function to enhance training efficiency. I used a learning rate scheduler to reduce the learning rate to fine-tune the training process periodically, helping to improve model performance.

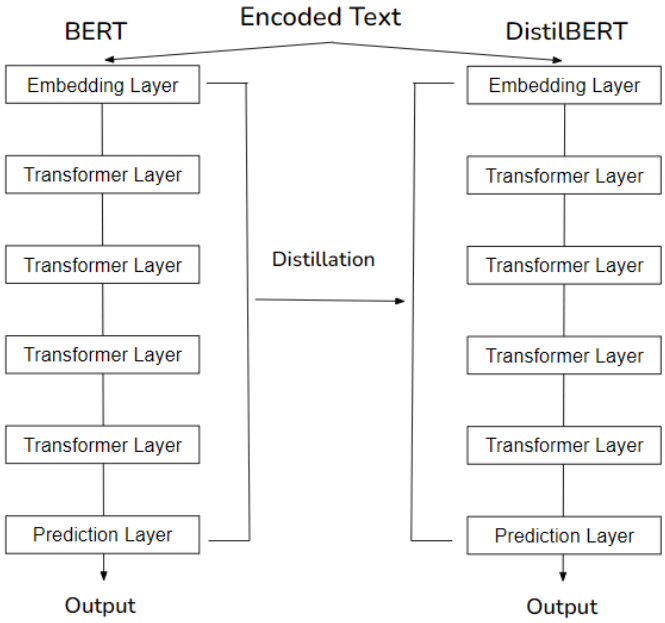


This neural network structure is commonly used for binary classification tasks due to its simplicity and effectiveness in capturing complex patterns in the data.

DistilBERT is a smaller, faster, and lighter version of the BERT model. DistilBERT retains most of BERT's natural language processing abilities while being more efficient. BERT (Bidirectional Encoder Representations from Transformers) is trained on Wikipedia and the BookCorpus. In total, it has been trained on around 3.3 billion words. This allows BERT to develop a highly comprehensive understanding of language.

The fine-tuning process involves adjusting the weights of DistilBERT's edges based on the dataset. We must also optimize for the binary cross-entropy loss function, which measures the difference between predicted and actual labels. The trainer class from the hugging face-transformers library manages data loading, optimization, and

evaluation. The compute_metrics function calculates the accuracy of the model's predictions. However, DistilBERT would take nearly a week to be trained on the entire 100,000 sample dataset. To ensure a rigorous and timely training procedure, I selected 10,000 samples at random and trained DistilBERT on this subset. This setup allows the model to leverage its pre-trained language knowledge and adapt to the specifics of sentiment analysis, resulting in a robust and efficient classifier.



Validation

Validation techniques such as confusion matrices[14] and cross-validation[15] are employed to ensure the accuracy of the model. These methods help assess the model's performance and ensure it generalizes well to unseen data.

A confusion matrix is a powerful tool for evaluating the performance of a classification model. It provides a detailed breakdown of the model's predictions compared to the actual labels. The confusion matrix is as follows:

True Positive (TP): Model outputs 1, Actual label 1	False Positive (FP): Model outputs 1, Actual Label 0
False Negative (FN): Model outputs 0 Actual label 1	True Negative (TN): Model outputs 0, Actual Label 0

Using these, it is possible to calculate several important metrics.

Accuracy, the overall percentage of correct predictions, calculated as

$$\frac{(TP + TN)}{(TP + FP + FN + TN)}.$$

Precision, the percentage of correct positive predictions, calculated as

$$\frac{(TP)}{(TP + FP)}.$$

Recall, the proportion of positives that are correctly identified, calculated as

$$\frac{(TP)}{(TP + FN)}.$$

F1 Score, the harmonic mean of precision and recall. It is calculated as

$$\frac{TP}{TP+0.5(FP+FN)}.$$

These metrics offer a comprehensive view of the model's performance and can be especially helpful when the data's class distribution is imbalanced.

Another validation method has to do with splitting the dataset. Instead of training and testing on one large dataset, we first divide the data into ten subsets called folds. Then, we use nine folds as the training set and the last fold as the validation set. We repeat this process ten times, using a different validation fold each time. This process is known as k-fold cross-validation. This helps ensure the model is not getting lucky in one fold. We average the accuracy across all the folds to find the final accuracy. This ensures that the accuracy is representative of the model's true performance.

To determine if the models run in a reasonable time, I created a dataset of 1000 samples and used the Python time module to average the individual predictions. This helps us ensure that the model does not take an absurdly long time to predict on a large dataset.

Using these techniques, we can guarantee our models' accuracy and reliability.

RESULTS

Character-based tokenization

Model	Accuracy	Loss
Decision Tree Classifier	52.41%	—

The model performed poorly due to the information deficit in character-based tokenization. It could not decipher the patterns of singular characters, resulting in an accuracy akin to blindly guessing.

Word-based tokenization

One-Hot Encoding

Model	Accuracy	Validation Loss	Learning Time (mins)	F1 Score	Inference Time (ms)
Decision Tree	54.87%	–	3	0.582	1
Random Forest	68.31%	–	43	0.741	4
Adaboost	70.73%	–	52	0.769	3

One-hot encoding suffers a similar problem as character-based tokenization, where there is simply a need for more information. In this case, the encoded vectors are large and sparse. This makes it difficult for the models to find patterns in the data as the vectors have too little variety.

Bag of Words Encoding

Model	Accuracy	Validation Loss	Learning Time (mins)	F1 Score	Inference Time (ms)
Decision Tree	62.87%	–	3	0.632	1
Random Forest	74.31%	–	43	0.741	6
Adaboost	79.44%	–	43	0.798	6
FNN	88.34%	0.638	126	–	9

DistilBERT	94.27%	–	526	–	36

Bag of Words encoding, on the other hand, adds more information into the encoded vectors and enables the models to achieve some strong predictions. However, it still suffers from many of the same issues as one-hot encoding.

TF-IDF

Model	Accuracy	Validation Loss	Learning Time (mins)	F1 Score	Inference Time (ms)
Decision Tree	63.31%	–	3	0.643	1
Random Forest	74.25%	–	36	0.745	6
Adaboost	80.04%	–	49	0.809	5
FNN	89.29%	0.712	176	–	9
DistilBERT	94.60%	–	533	–	38

Although TF-IDF mitigates most of BoW's flaws, there is only a slight increase in accuracy. This could indicate that the accuracy is limited by the model design or dependent on factors outside of encoding.

Other Datasets

Model	Accuracy - AI Data	Accuracy - Human Data
Adaboost	79.56%	80.12%
FNN	89.34%	88.79%
DistilBERT	94.78%	94.45

There was no significant difference in the model's ability to predict formal datasets, my dataset, or the ChatGPT dataset.

DISCUSSION

A fine-tuned version of DistilBERT is the strongest model. It achieves a 94.89% accuracy which is extremely accurate and can be used in almost every task. Since I plan to implement a model to scan large libraries of texts and messages, it would be best to use the most accurate model. The main drawback of DistilBERT is its high inference and learning times. Learning time is the time it takes to train the model. It is not a big issue as the model only needs to be trained once. It is only inconvenient when I update the model because I have to re-train it. To minimize this issue regardless, I recommend decreasing the number of epochs or the size of the dataset. Inference time, on the other hand, matters a lot when dealing with large amounts of data. To predict a dataset of a billion messages, the DistilBERT model would require 440 days. However, a decision tree would need only 11 days to finish the same task. Even so, the high accuracy makes the trade-off worth it, as it eliminates the need for manual verification. Fortunately, the model will never be used on a dataset that large.

The decision tree classifier stands out as the most intuitive model. However, it is difficult to make many meaningful splits without overfitting. Therefore, it's highly inaccurate, and I would not recommend using it. For smaller tasks, I recommend the neural network or the Adaboost algorithm. Both have sufficient accuracy, and a small use case would ensure that it is possible to verify each prediction. To improve Adaboost, I have found that using the algorithm on the predictions of a random forest works significantly better than that of a sole decision tree. This is due to the complexity of the data, as a single decision tree is too weak for a learner. I recommend changing the size and number of layers to improve the neural network. Currently, it is prone to overfitting on the training dataset, and reducing the complexity of the neural network slightly could improve its validation loss score.

CONCLUSION

Among the many models tested in this paper, DistilBERT stands out as the most accurate. To be used on a massive collection of text, increasing its accuracy will reduce the need for human verification. By leveraging DistilBERT’s advanced capabilities for sentiment analysis, we can more effectively detect signs of mental health issues from digital communication and raise awareness. The California Department of Healthcare Services finds raising awareness can reduce the stigma associated with receiving treatment for mental illnesses[16]. I hope to aid many of the 280 million people affected by depression by bringing attention and directing support to those in need.

NEXT STEPS

Although movie reviews are a good indicator of emotion, meaning might be

sometimes concealed. In particular, a text might be sarcastic, which is difficult for the machine learning models to handle. Furthermore, the degree of emotion also varies drastically with every message, so a slightly negative text might be processed differently from an extremely negative text. The challenge is ensuring the model can discern the text's true meaning. To do so, the model has to be trained again to understand sarcasm, slang, and the other complexities of our language.

ACKNOWLEDGEMENTS

I would like to thank my mentor, Jason Liang, for guiding me through the research process and my family for supporting me throughout.

REFERENCES

1. National Library of Medicine, *Child and Adolescent Mental Health*. Agency for Healthcare Research and Quality (US), 2022. Available: https://www.ncbi.nlm.nih.gov/books/NBK587174/
2. R. Socher *et al.*, "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," *ACL Anthology*, pp. 1631–1642, Oct. 2013, Accessed: Jun. 18, 2024. [Online]. Available: https://www.aclweb.org/anthology/D13-1170
3. A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," *ACLWeb*, Jun. 01, 2011. http://www.aclweb.org/anthology/P11-1015 (accessed Jun. 17, 2024).
4. F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011, Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
5. J. Ansel *et al.*, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," Apr. 2024, doi: https://doi.org/10.1145/3620665.3640366.
6. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a Distilled Version of BERT: smaller, faster, Cheaper and Lighter," *arXiv.org*, 2019. https://arxiv.org/abs/1910.01108
7. American Academy of Pediatrics, "AAP-AACAP-CHA Declaration of a National Emergency in Child and Adolescent Mental Health," *www.aap.org*, Oct. 19, 2021. https://www.aap.org/en/advocacy/child-and-adolescent-healthy-mental-development/aap-aacap-cha-declaration-of-a-national-emergency-in-child-and-adolescent-mental-health/ (accessed Jul. 18, 2024).
8. National Institute Of Mental Health, "Depression," *National Institute of Mental Health*, Mar. 2023. https://www.nimh.nih.gov/health/topics/depression (accessed Jul. 23, 2024).
9. W. Uther *et al.*, "TF–IDF," *Encyclopedia of Machine Learning*, pp. 986–987, 2011, doi: https://doi.org/10.1007/978-0-387-30164-8_832.
10. P. Whatman, "Social Sentiment Online: Yes, Social Media is Becoming More Negative," Mention, Mar. 27, 2018. https://mention.com/en/blog/social-media-mentions-analysis (accessed Aug. 19, 2024).
11. S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. Beijing Etc.: O'reilly, 2009.
12. D. Berrar and W. Dubitzky, "Information Gain," *Springer eBooks*, pp. 1022–1023, Jan. 2013, doi: https://doi.org/10.1007/978-1-4419-9863-7_719.
13. Y. Dodge, "Gini Index," *The Concise Encyclopedia of Statistics*, pp. 231–233, 2021, doi: https://doi.org/10.1007/978-0-387-32833-1_169.
14. T. R. Shultz *et al.*, "Confusion Matrix," *Encyclopedia of Machine Learning*, pp. 209–209, 2011, doi: https://doi.org/10.1007/978-0-387-30164-8_157.
15. P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," *Encyclopedia of Database Systems*, pp. 532–538, 2009, doi: https://doi.org/10.1007/978-0-387-39940-9_565.
16. "Mental Health Awareness," Ca.gov, 2023. https://www.dhcs.ca.gov/services/MH/Pages/MHAM_Matters.aspx