



Trigger Warning: Mentions of suicide

Early Detection and Intervention of Suicide using sentiment analysis



Mental Health Crisis : According to the World Health Organization (WHO) close to 800,000 people die by suicide each year, making it a significant public health concern.

Underreporting and Stigma : Mental health issues leading to suicide are often underreported due to social stigma and barriers to accessing mental health services. Sentiment analysis can act as an tool to gauge emotional distress from online textual content, providing insights into the mental well-being of individuals who might not otherwise seek help.

Early Detection and Intervention : Timely detection of depression and suicidal tendencies is critical for saving lives.

Suicidal Comments Detection

Contributions

- **Erin:** EDA, Sentiment Analysis + Modeling, Conclusion, Limitations/Improvements, Social Impact
- **Jordan:** Pre-processing (Slang & Spellcheck), Embeddings + Modeling, Model Blending
- **Shaki:** Pre-processing (Text filtering, Lemmatization, POS tagging & Chunking/Chinking), TF-IDF + Modeling

Jordan Fan
Shaki Pothini
Erin Smith

W207 Summer 2023
August 8, 2023

Overview Suicide and Depression Detection Dataset

Source : [Kaggle Suicide and Depression Detection](#)

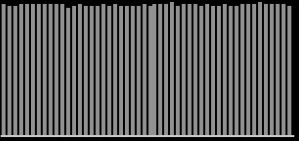
Description:

The dataset comprises posts from Reddit subreddit: "SuicideWatch" along with non-suicide posts from the "r/teenagers" subreddit.

Posts from "SuicideWatch" were collected from Dec 16, 2008 (creation) to Jan 2, 2021

All posts collected from "SuicideWatch" are labeled as "suicide," and those from Non-suicide posts collected from "r/teenagers." are labelled as 'non-suicide

Overview Suicide and Depression Detection Dataset ...

#	text	class
	context extracted from user posts(Input)	target variable
 2 348k	232074 unique values	2 unique values
2	Ex Wife Threatening SuicideRecently I left my wife for good because she has cheated on me twice and ...	suicide
3	Am I weird I don't get affected by compliments if it's coming from someone I know irl but I feel rea...	non-suicide

Data Model :

Two columns

- “text” : user posts in string
- “Class” : target variable in string

Two types of classification for the target class :

- “suicide”,
- “non-suicide”

348k rows in the raw data set

Assumptions / Hypotheses

Sentiment as a Predictive Indicator : We assume that sentiments expressed in textual data can serve as valuable predictive indicators of an individual's emotional well-being. Analyzing sentiments can help uncover signs of depression or suicidal tendencies, facilitating early intervention.

Context based model : Contextual understanding of language is crucial for detecting subtle nuances in sentiment. Our approach would be analyze text by following methods and then build models like RandomForest, Logistic Regression, Gradient Boosting, CNN etc.

1. **TF-IDF:** Numerical representation of word importance in a document compared to a collection of documents.
2. **VADER:** Rule-based sentiment analysis tool for social media text, considering valence and intensity for sentiment scores.
3. **Word Embeddings:** Represent words as dense vectors, capturing semantic relationships based on context.
4. **Doc Embeddings:** Represent entire documents as fixed-dimensional vectors, capturing overall context and meaning.

Model Blending for Enhanced Performance: Combining the predictions from multiple models will lead to higher accuracy compared to using any individual model in isolation. Model blending leverages the strengths of various models, reducing biases and errors, resulting in a more robust suicide detection system.

Pre-Processing - Slang & Spell Check

1. Slang Translation

- Dictionary of slang words, acronyms, emoticon to meaning from Wordpress

2. Spell check

- SymSpell library - fast and able to split words missing spaces

3. Contraction

- split contracted words to their uncontracted form
- ex: couldn't -> could not

Slang	Meaning	
>:(Angry	
>:)	Evil Grin	
GLOCK	Semi-automatic pistol	
GURL	Girl	
GUD	Good	
GTA	Grand Theft Auto	
ILY	I Love You	
L8R	Later	

Original Text:

Anybody wanna play FlingSmash? I'm bored

Cleaned Text:

anybody want to play fling smash i am bored

Pre-Processing: stop words, punctuations, lemmatization

4. **Remove stop words** are common words (e.g., 'the', 'such', 'than', 'this', 'up', 'other', 'about') removed from text analysis due to their limited significance

- download stop words from nltk for english `162 stop words removed`
- identify words to exclude like no, don't etc.

5. **Remove punctuations** anything that is not alphanumeric or white space : `str.replace(r'[^\w\s]', '')`

6. **Lemmatization** : Reducing words to their base form (lemma) while considering part of speech for accuracy.

- WordNet data, is a lexical database used for lemmatization in NLTK.
- Lemmatizes each word in the tokenized list using the WordNetLemmatizer

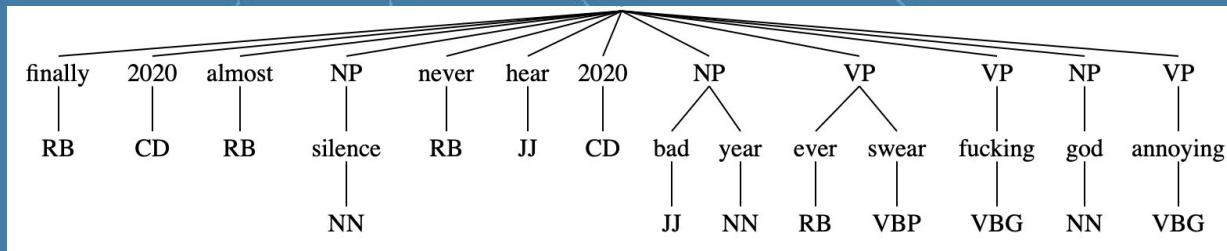
INPUT: "[Trigger warning] Excuse for self inflicted burns*I do know the crisis line and used it after when I was having a panic attack. *I know it's not a healthy thing to do."

OUTPUT: trigger warning excuse self inflicted burn i know crisis line used panic attack know not healthy thing

Pre-Processing : POS and Chunking/Chinking

7. **Chunking and Chinking**: We want to extract/chunk noun phrases and verb phrases and chink/exclude conjunctions and prepositions.

- First step is to perform **Parts of Speech (POS)** Tagging. Used the `nltk.download('averaged_perceptron_tagger'` for POS tagging data.



- Used the following rule for chunking and chinking

`chunk_grammar = r"""`

`NP: {<DT>?<JJ>*<NN.*>}`

`VP: {<RB.??>*<VB.*><RB.??>*}`

`><CC|IN>{`

`"""`

- Noun (NN): Represents a noun, e.g., "dog," "cat," "table."
- Verb (VB): Represents a verb, e.g., "run," "eat," "sleep."
- Adjective (JJ): Represents an adjective, e.g., "happy," "beautiful," "tall."
- Adverb (RB): Represents an adverb, e.g., "quickly," "eagerly," "slowly."
- Pronoun (PRP): Represents a pronoun, e.g., "he," "she," "it," "they."
- Preposition (IN): Represents a preposition, e.g., "in," "on," "at," "with."

INPUT: "finally 2020 almost silence never hear 2020 bad year ever swear fucking god annoying"

OUTPUT: "silence bad year ever swear fucking god annoying"

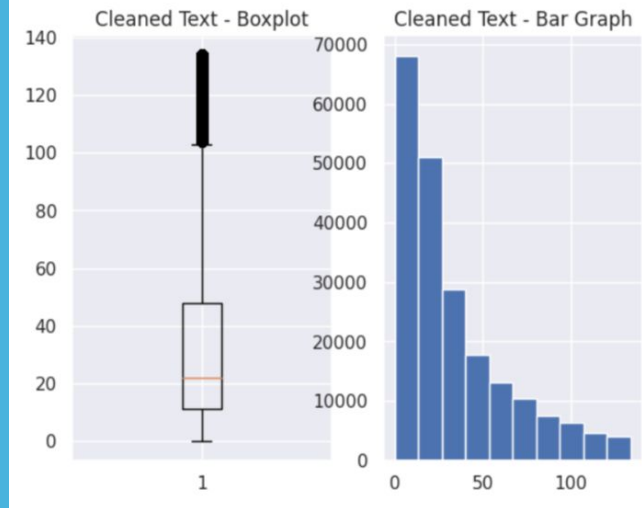
Data Exploration – Text Length

- Text Length Distribution
 - Cleaned vs Raw Text: text shortens after cleaning
 - Outliers: A few very long texts heavily skewed data

	len_text	len_text_cleaned
count	232073.000000	232073.000000
mean	131.925730	52.988202
std	217.477124	90.226019
min	1.000000	0.000000
25%	26.000000	12.000000
50%	60.000000	25.000000
75%	155.000000	61.000000
max	9684.000000	5850.000000

```
Old Shape: (232073, 7)
49.0
Upper Bound: 134.5
20530
Lower Bound: -61.5
0
New Shape: (211543, 7)
```

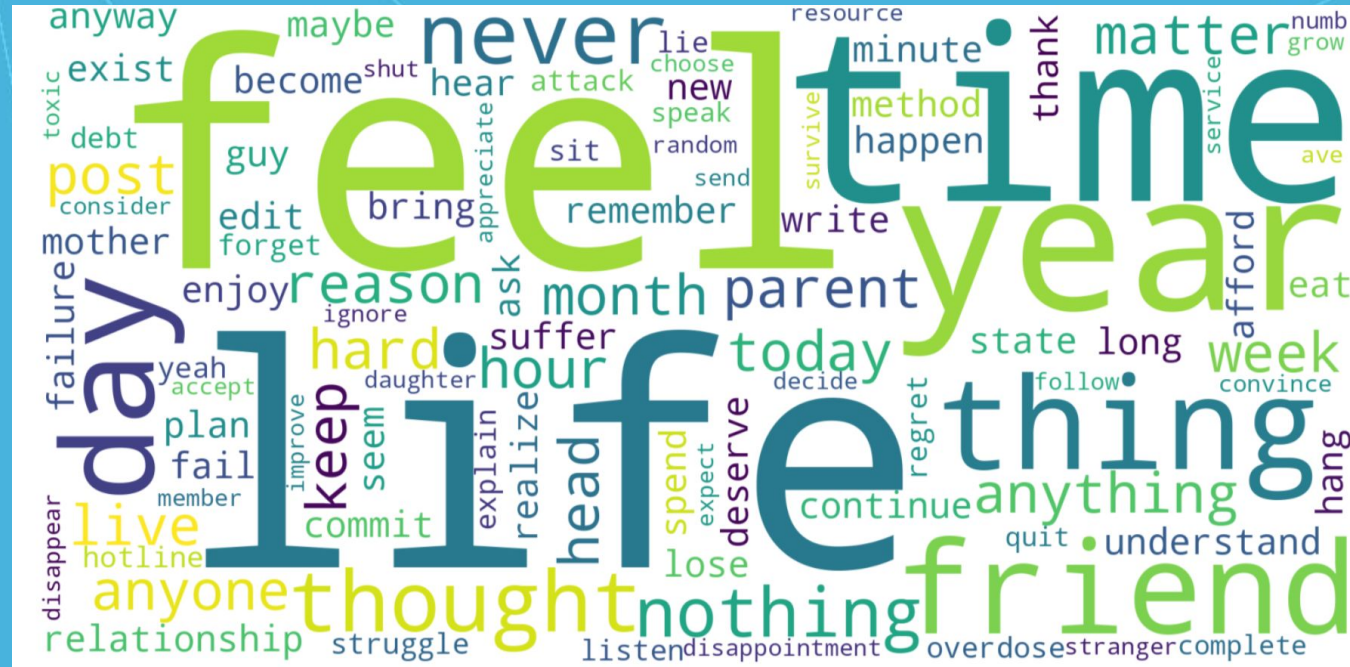
```
count    211543.000000
mean       33.894192
std        30.632028
min         0.000000
25%        11.000000
50%        22.000000
75%        48.000000
max       134.000000
Name: len_text_cleaned, dtype: float64
```



Data Exploration – Word Frequency

Suicide Class

	word	count
13	want	72357
273	feel	56150
42	life	54696
100	know	51341
31	get	39215



Non-Suicide Class

Feature Engineering – Sentiment Analysis

- **Hypothesis:** Binary sentiment detection models can be extended Binary suicide detection models
- **Developed sentiment features** such as probability score and binary sentiment classification
- **Baseline:** Binary sentiment classification = Suicide label
 - If sentiment is pos, then label is non-suicidal
 - If sentiment is neg, then label is suicidal
- **Logistic Regression Model** based on sentiment probability score

Models + Results – Sentiment Analysis

Baseline Accuracy: 0.37

Logistic Regression	precision	recall	f1-score
non-suicide	.50	.48	.49
suicide	.50	.53	.52

	text	class	text_cleaned	len_text	len_text_cleaned	pos_tags	chunk_chink	text_sentiment_score	text_sentiment_prob	text_sentiment
0	Ex Wife Threatening SuicideRecently I left my ...	suicide	former wife threatening suicide recently left ...	715	378	[(former, JJ), (wife, NN), (threatening, VBG),...	[[(former, JJ), (wife, NN)], [(threatening, VB...	{'neg': 0.444, 'neu': 0.429, 'pos': 0.127, 'co...	-0.9827	neg
1	Am I weird I don't get affected by compliments...	non-suicide	weird get affected compliment coming someone k...	138	94	[(weird, JJ), (get, NN), (affected, JJ), (comp...	[[(weird, JJ), (get, NN)], [(affected, JJ), (c...	{'neg': 0.168, 'neu': 0.51, 'pos': 0.321, 'com...	0.6115	pos
2	Finally 2020 is almost over... So I can never ...	non-suicide	silence bad year ever swear fucking god annoying	129	48	[(finally, RB), (2020, CD), (almost, RB), (sil...	[[(finally, RB), (2020, CD), (almost, RB), [(si...	{'neg': 0.546, 'neu': 0.284, 'pos': 0.17, 'com...	-0.6469	neg
3	i need helpjust help me im crying so hard	suicide	need help help cry	41	18	[(need, NN), (help, NN), (help, VB), (cry, VB)...	[[(need, NN)], [(help, NN)], [(help, VB)], [(c...	{'neg': 0.326, 'neu': 0.105, 'pos': 0.568, 'co...	0.3182	pos
4	I'm so lostHello, my name is Adam (16) and I'v...	suicide	ism lost hello name struggling year ism past y...	2426	1369	[(ism, NN), (lost, VBD), (hello, JJ), (name, N...	[[(ism, NN)], [(lost, VBD)], [(hello, JJ), (na...	{'neg': 0.368, 'neu': 0.535, 'pos': 0.097, 'co...	-0.9969	neg

Feature Engineering : TF-IDF

Term Frequency : measures how frequently a word appears in a document.

Inverse Document Frequency : measures how rare or unique a word is across all documents in the corpus. TF-IDF score gives higher weight to words that are both frequent in the document (high TF) and rare across the corpus (high IDF).

Hypothesis : behind using TF-IDF as feature representation technique for sentiment analysis is that certain words or terms carry more significance in expressing sentiment than others

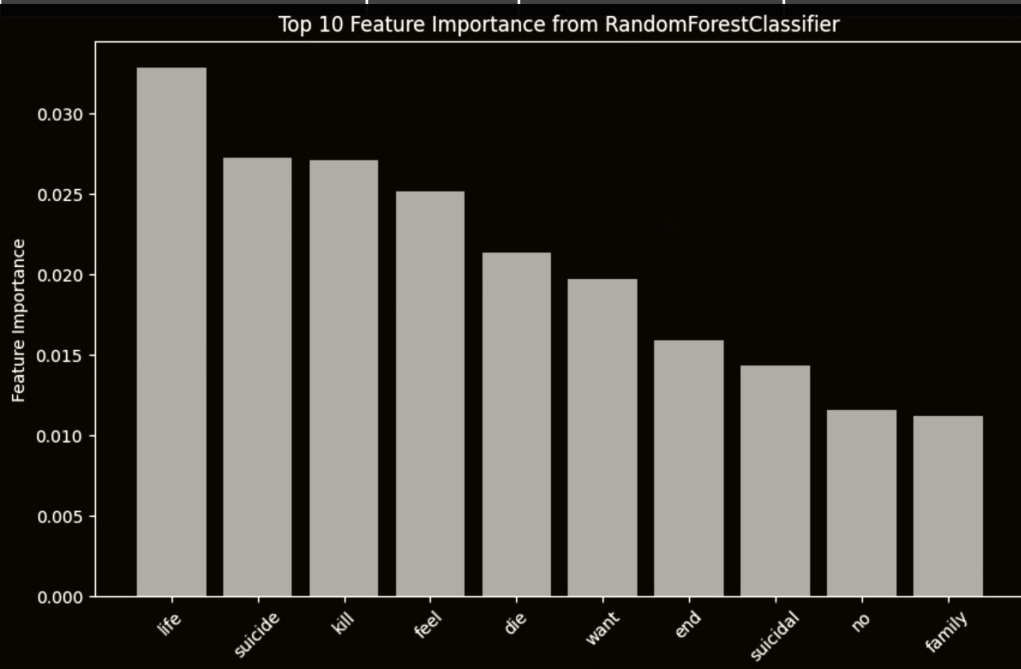
Baseline model : Gradient Boosting as sentiment classifier, the model will start with a simple model and then iteratively add more models. Each new model focuses on the documents that the previous models struggled to classify correctly.

Random Forest: uses the TF-IDF representations of labeled documents to build multiple decision trees. Each tree is trained on a different subset of the data to predict the sentiment labels based on TF-IDF score.

Hyper parameter split Criterion : Gini impurity measures the probability of misclassifying a randomly chosen data point label within a node. Entropy measures the amount of uncertainty or disorder within a node where 0 represents a pure node (all data points belong to the same class) and 1 represents maximum uncertainty (data points are evenly split between classes).

Hyper parameter max_features : determines the maximum number of features (words in the case of TF-IDF data) to consider when looking for the best split at each node of the decision tree. 'None', 'sqrt', 0.2

MODEL	ACCURACY	PRECISION (non-suicide)	RECALL (non-suicide)	F1 (non-suicide)	PRECISION (suicide)	RECALL (suicide)	F1 (suicide)
GRADIENT BOOSTING	0.87	0.84	0.92	0.88	0.91	0.83	0.87
RANDOM FOREST criterion='gini', max_features=None	0.90	0.91	0.88	0.90	0.89	0.91	0.90
RANDOM FOREST criterion='entropy', max_features='sqrt'	0.90	0.90	0.89	0.90	0.90	0.91	0.90
RANDOM FOREST criterion='entropy', max_features=0.2	0.89	0.89	.90	.89	0.89	0.89	0.89



Feature Importance – Keywords:

life, suicide, kill, feel want, end,
suicidal, no, family

Feature Engineering – Word Embeddings

- **Embedding Preprocessing**
 - Padded text to 100 words
 - Limited vocabulary to 5k words
- **Average Pooling Word Embeddings**
 - Tuned: Vocabulary size, embedding output dimension, # hidden layers / # nodes

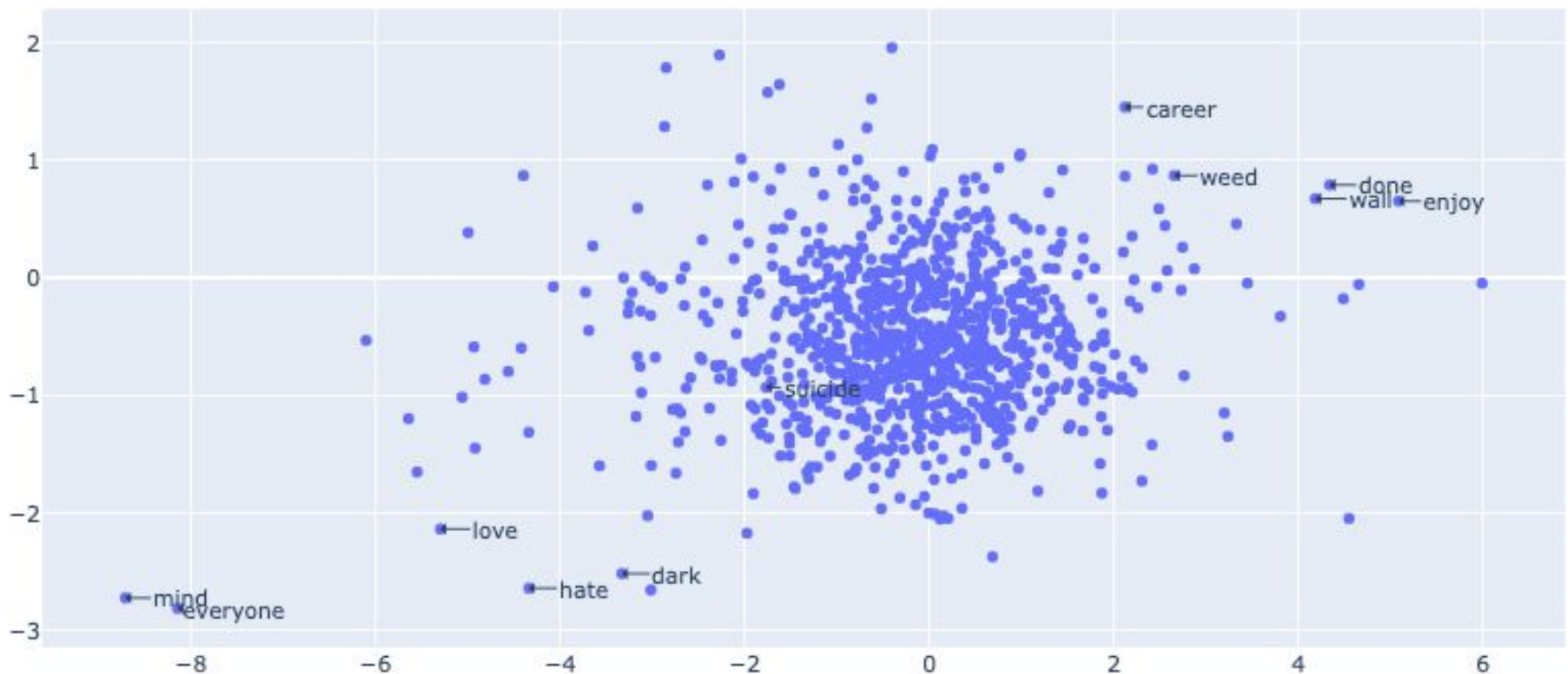
Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 100)	484500
global_average_pooling1d (GlobalAveragePooling1D)	(None, 100)	0
dense (Dense)	(None, 100)	10100
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 50)	5050
dropout_1 (Dropout)	(None, 50)	0
dense_2 (Dense)	(None, 1)	51
Total params: 499701 (1.91 MB)		
Trainable params: 499701 (1.91 MB)		
Non-trainable params: 0 (0.00 Byte)		

- **1D Convolutional Neural Network Embeddings**
 - Tuned: # filters, kernel size, # hidden layers

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 100)	484500
conv1d (Conv1D)	(None, None, 128)	51328
dropout (Dropout)	(None, None, 128)	0
max_pooling1d (MaxPooling1D)	(None, None, 128)	0
conv1d_1 (Conv1D)	(None, None, 64)	32832
dropout_1 (Dropout)	(None, None, 64)	0
max_pooling1d_1 (MaxPooling1D)	(None, None, 64)	0
conv1d_2 (Conv1D)	(None, None, 32)	10272
dropout_2 (Dropout)	(None, None, 32)	0
max_pooling1d_2 (MaxPooling1D)	(None, None, 32)	0
global_average_pooling1d (GlobalAveragePooling1D)	(None, 32)	0
dense (Dense)	(None, 1)	33
Total params: 578965 (2.21 MB)		
Trainable params: 578965 (2.21 MB)		

Feature Engineering – Word Embeddings

Word Embeddings



Feature Engineering – Document Embeddings

- **Document Embeddings**
 - Gather additional context from entire document instead of word-based
 - Doc2Vec library: Limit to 100 words and output 1D array of size 100 (similar to word embedding output)

```
#Create similar embedding size and minimum word count as word embedding model  
d2v = gensim.models.doc2vec.Doc2Vec(vector_size=100, min_count=100, epochs=20)  
d2v.build_vocab(train_tagged)  
d2v.train(train_tagged, total_examples=d2v.corpus_count, epochs = d2v.epochs)
```

```
#Create document sentence based embeddings  
d2v_train = np.array([d2v.infer_vector(doc.words) for doc in train_tagged])  
d2v_val = np.array([d2v.infer_vector(doc.words) for doc in val_tagged])  
d2v_test = np.array([d2v.infer_vector(doc.words) for doc in test_tagged])
```

Models – Embeddings

- Logistic Regression and Neural Network was run on document embeddings
- Processed word embedding and combined with document embedding to pass into Gradient Boosted Trees
 - Translated each word to respective array in word embedding, took average across embedding index
 - Combined embeddings in case embeddings captured different information or trends
 - LightGBM: grow boosted trees leaf first instead of level first - able to train faster
 - Hyperparameters tuned: # leaves, L1 regularization

Models Results – Embeddings

Model	Accuracy	Non-Suicide Precision	Non-Suicide Recall	Non-Suicide F1	Suicide Precision	Suicide Recall	Suicide F1
Avg Pooling Word Embedding	0.925	0.925	0.925	0.925	0.925	0.924	0.924
1D CNN Word Embedding	0.921	0.934	0.907	0.92	0.909	0.936	0.922
Doc2Vec Logistic Regression	0.846	0.813	0.9	0.854	0.887	0.792	0.837
Doc2Vec Neural Network	0.884	0.876	0.895	0.885	0.892	0.873	0.883
Boosted Trees w/ Reg Word Embedding	0.925	0.927	0.924	0.925	0.924	0.927	0.925
Boosted Trees w/ 1D CNN Word Embedding	0.925	0.927	0.924	0.925	0.924	0.927	0.925

Model Blending

- Separate and diverse views of the data
 - Models generated from different perspectives could potentially be better at predicting some scenarios better than others
- Trained and tuned models on training set, use final models to output predicted probabilities on hold out validation and test set
- Validation set used to train on predicted probabilities to determine how to weight each output
- Model: Gradient Boosted Trees
 - Improve upon errors of each model

Model Blending Results

	precision	recall	f1-score	support
0	0.933	0.933	0.933	23248
1	0.932	0.933	0.932	23167
accuracy			0.933	46415
macro avg	0.933	0.933	0.933	46415
weighted avg	0.933	0.933	0.933	46415

Conclusion

- **Key Results**

- Context-based suicide detection models can be developed with high accuracy (~90%)
- TFIDF or Embedding models are better than sentiment-based models (~40%)
- Model blending does improve accuracy (~3%)

- **Practical Significance**

- This model can help suicide prevention organizations identify and intervene with individuals who may be at risk of suicide before it happens.

Limitations & Future Improvements



Data Processing

Suicide label
Non-Suicide label



Data Collection

Generalizability
Bias

Social Impact Considerations

Predictive Policing

False Negatives



References

1. **Internet Slang Dataset** - <https://floatcode.wordpress.com/2015/11/28/internet-slang-dataset/>
2. **SymSpell example** - https://symspellpy.readthedocs.io/en/latest/examples/lookup_compound.html
3. **Natural Language Preprocessing** - <https://towardsai.net/p/nlp/natural-language-processing-nlp-with-python-tutorial-for-beginners-1f54e610a1a0#01f7>
4. **Fine-Tuning Random Forest Guide** - <https://towardsdatascience.com/random-forest-hyperparameters-and-how-to-fine-tune-them-17aee785ee0d>
5. **Random Forest Feature Importance** - https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
6. **Hyperparameter Tuning Random Forest** - <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>
7. **Hyperparameter Tuning LightGBM** - <https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#deal-with-over-fitting>
8. **Tensorflow Early Stopping** - https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
9. **Flattening Word Embeddings** - <https://stackoverflow.com/questions/67381956/how-do-we-use-a-random-forest-for-sentence-classification-using-word-embedding>