Can the suicidal intentions of young people be predicted using the language analysis of Reddit posts?

I. Problem Description

The dataset is "Reddit dataset for Multi-task NLP" on Kaggle with 226953 samples.

1. Summary of input data and output data

Input features for four models (KNN, Logistic Regression, SGD, and RNN) are word embeddings derived from the preprocessed textual data. The models learn from these embeddings to perform the binary classification task (suicidal or non-suicidal) on new, unseen text data. The output of each model is the predicted label for each input instance, which indicates whether the text is classified as suicidal (1) or non-suicidal (0).

2. Attributes of key features for analysis/prediction

Model	Key features
K-Nearest Neighbors (KNN)	- The prediction for a data point is based on the majority class of its K nearest neighbors. - Key features are words that are commonly present in the posts of the K nearest neighbors
Logistic Regression (LR)	- LR coefficients represent the weight assigned to each feature (word) in predicting the output. - Key features are words with higher absolute weights
Stochastic Gradient Descent (SGD) Classifier	- SGD assigns weights to each feature (word) based on their importance in predicting the output. - Key features are words with higher absolute weights
Recursive Neural Network (RNN)	- RNN uses the attention mechanism to indicate the relevance of each word to the final prediction. - Key features are words with higher attention weights

Table 1: Data features and description

II. Data Preprocessing

Data preprocessing cleans and standardizes the text data, remove noise, handle missing values, and convert text into numerical representations using word embeddings.

1. Imputation

The 'Post' column in the dataset contains missing values. These missing values are filled with the string 'default_text' to ensure all instances have valid data.

2. Feature removal

re.sub('[^a-z]+', ' ', phrase, flags = re.IGNORECASE)	Remove all non-alphabetic characters, keeping only alphabetic characters in lowercase: focus solely on the semantic meaning of words
re.sub('(\s+)', ' ', phrase)	Replace extra whitespaces with a single whitespace: ensure uniformity and consistency in the text data, preventing issues with tokenization and word embeddings
re.sub('http\S+', ' ', phrase)	Remove URLs: eliminate non-essential information
lower()	Convert the text to lowercase: preventing case-specific distinctions in language
list(stopwords.words()) + ['filler']	Remove stop words: reduce noise and unnecessary words that do not carry significant meaning

Table 2: Feature removal processing and description

3. Pre-processing transformations

- Word embedding and word averaging: Pre-trained FastText model converts a sequence of words into a fixed-size vector. It calculates the mean vector of all word vectors in a text to obtain a single vector representation for the input text. The advantage of this step is capture the overall context and semantics of the text by considering the combined influence of all the words present in the text, rather than just focusing on individual words.
- Tokenization: When tokenizing text with NLTK library, the input text is split into separate words, removing any punctuation, special characters, or spaces.

III. Model Selection

Table 3 and 4 figure out the strengths and weaknesses of each model. Overall, KNN and Logistic Regression are classic models known for their simplicity and effectiveness, making them suitable for the initial exploration of the data. SCD classifier works better with non-linear relationship in the data compared to KNN and Logistic Regression. The RNN model, on the other hand, is a more sophisticated approach that can effectively capture the contextual information present in text data.

Model	Strength	Weakness
K-Nearest Neighbors (KNN)	- KNN is one of the simplest algorithms and easy to understand. - KNN does not assume any specific data distribution, and not requiring complex preprocessing.	- KNN needs to compute distances to all data points in the training set, making it time-consuming for large datasets. - KNN can be affected by noisy data points and boundaries of classes, leading to inaccurate classification decisions.
Logistic Regression (LR)	- LR is simple and easy to implement. It does not require many computational resources, saving time in training and often gives good results for binary classification tasks. - LR provides coefficients for each feature, helping understand the impact of each feature on the classification result.	- LR cannot handle complex and nonlinear relationships between features and the output, leading to inaccurate predictions. - Outliers can significantly affect the LR model, reducing its stability and generalization ability.

Table 3: Comparison of strength and weakness of KNN and logistic regression

Model	Strength	Weakness
Stochastic Gradient Descent (SGD) Classifier	- SGD is an optimization algorithm that updates model weights based on individual data points quickly and work on large datasets with good performance. SGD can handle non-linear relationships between features and the output.	SGD model cannot capture or mimic sequential relationships in text data as an RNN model does, resulting in missing relevant contextual information and word order within sentences or text.
Recursive Neural Network (RNN)	- RNN handles sequential data like text and retain information from previous parts and use it to predict the next parts, capturing the sequential relationship between words or sentences, the meaning of words in the context of suicide texts. - RNN learns complex patterns in data, especially when there are long-term dependencies between words or sentences in the text.	RNN requires large and diverse data to effectively learn complex patterns. Insufficient or insufficiently diverse data can lead to overfitting.

Table 4: Comparison of strength and weakness of SGD classifier and RNN

IV. Model Refinement

1. Feature engineering

The FastText pre-trained word embeddings converts the text data into numerical vectors and the word averaging technique to calculate the mean of word embeddings for each text post.

2. Sample splitting

- Training Set: 60% of the original data (used for training the model).
- Validation Set: 20% of the original data (used for hyperparameter tuning and model evaluation during training).
- Test Set: 20% of the original data (used for final evaluation of the trained model's performance).

3. Hyperparameter adjusting

Model	Hyperparameters
K-Nearest Neighbors (KNN)	- “n_neighbors” in [3, 5, 7, 10]: the number of neighbors - “weights” in [‘uniform’, ‘distance’]: test the contribution of members of the neighborhood via different weightings (weights)
Logistic Regression	- “c” in [0.001, 0.01, 0.1, 1, 10, 100]: control the amount of regularization applied to the model - “solver” in ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']: determine the optimization algorithm
Stochastic Gradient Descent (SGD) Classifier	- “alpha” in [0.000001, 0.00001, 0.0001, 0.001, 0.01]: the learning rate towards the optimal solution - “penalty” in ['l2', 'l1', 'elasticnet']: the type of regularization
Recursive Neural Network (RNN)	- Dense Layer Size: a dense layer with 64 neurons - Activation function: ReLU introduces non-linearity and helps the model learn complex relationships between input and output. The model uses a binary classification setup, so a single neuron with sigmoid activation is used for the output layer. - Dropout rate: 0.5 (50% of the neurons will be randomly dropped out during training) - Epochs and batch size: 100 epochs with a batch size of 32 - Early stopping callback: the training will stop and restore the weights with the best validation score after 5 consecutive epochs not improved

Table 5: Hyperparameters of the models and description

4. Model building and cross-validation

- GridSearchCV is the hyperparameter tuning method for KNN, logistic regression and SGD.
- GridSearchCV performs 5-fold cross-validation to find the best combination of hyperparameters that optimize the model's performance.
- Training an RNN model with GridSearchCV is computationally intensive and time-consuming. Instead, we manually choose specific hyperparameters and use KFold cross validation for the RNN model.

5. Evaluation

- Recall metric is used as evaluation metrics to measure the model's overall performance and its ability to correctly classify both classes (suicidal and non-suicidal).

6. Model refinement

- Text preprocessing: experiment using lemmatization and stemming
- Word embedding models: use different models like Word2Vec, GloVe
- Hyperparameter adjusting:

  + K-Nearest Neighbors (KNN): Add “metric” in [‘euclidean’, ‘manhattan’, ‘minkowski’] to test different distance metrics (metric) for choosing the composition of the neighborhood

  + Logistic regression: Try different regularization techniques “penalty” in [‘none’, ‘l1’, ‘l2’, ‘elasticnet’] to work with parameter “c”

  + SGD: try different learning rate schedules (e.g., 'constant', 'adaptive')

  + RNN: try different dropout rate at 0.2, add more layers and use the hyperparameter tuning method
- Evaluation metrics: consider the balanced precision and recall trade-off

V. Performance Description

For four models, the evaluation metric used is the “Recall” metric. Recall is the proportion of actual positive cases correctly identified by the model out of all true positive cases.

Recall is an essential metric for the task of suicide risk prediction because it measures the model’s ability to correctly identify individuals who are actually at risk of suicide. Maximizing recall ensures that the model can detect positive cases (individuals at risk) and minimizes the risk of missing potential suicide cases.

The model that achieves the highest recall on the test set is considered the best performing model. However, achieving high recall might come at the cost of reduced precision, as the model may be more prone to making false positive predictions, so we must consider the balanced precision and recall trade-off to fairly compare chosen models.

VI. Results Interpretation

K-Nearest Neighbors (KNN)	`Best Recall: 0.826203 using {'n_neighbors': 10, 'weights': 'uniform'}`
Logistic Regression	`Best Recall: 0.896724 using {'C': 100, 'solver': 'newton-cg'}`
Stochastic Gradient Descent (SGD) Classifier	`Best Recall: 0.897947 using {'alpha': 1e-05, 'penalty': 'l1'}`
Recursive Neural Network (RNN)	`Recall: 91.71% using {'Dense': (64, input_shape=(300,), 'Activation': 'relu', 'Dropout': 0.5, 'Dense': (1, activation='sigmoid'), 'loss': 'binary_crossentropy', 'optimizer': 'adam', 'early_stopping'}`

Table 6: Recall scores of the models with the best parameters

Based on the recall metric, the RNN model is the most appropriate for selection because it has the highest recall score outperforming the other models in correctly identifying individuals at risk of suicide.

Hyperparameters including a single hidden layer containing 64 neurons using the ReLU activation function, the sigmoid activation for the output layer, a dropout rate of 0.5, 100 epochs with a batch size of 32 are good choices for RNN.

About intrinsic model qualities, RNN effectively captures the sequential nature of textual data, the context and meaning of words in a sentence to understand the nuances of language used in suicide-related posts and identify signs of suicide risk accurately.

Based on the precision score from the classification reports (figure 1 and 2), RNN is still the most appropriate choice. It achieves the highest recall score while maintaining a balanced precision and recall trade-off.

KNN

	precision	recall	f1-score	support
0	0.94	0.70	0.80	22684
1	0.76	0.96	0.85	22707
accuracy			0.83	45391
macro avg	0.85	0.83	0.83	45391
weighted avg	0.85	0.83	0.83	45391

Logistic Regression

	precision	recall	f1-score	support
0	0.91	0.88	0.90	22684
1	0.89	0.92	0.90	22707
accuracy			0.90	45391
macro avg	0.90	0.90	0.90	45391
weighted avg	0.90	0.90	0.90	45391

Figure 1: Classification reports of KNN and logistic regression

SGD Classifier

	precision	recall	f1-score	support
0	0.91	0.89	0.90	22684
1	0.89	0.91	0.90	22707
accuracy			0.90	45391
macro avg	0.90	0.90	0.90	45391
weighted avg	0.90	0.90	0.90	45391

RNN

	precision	recall	f1-score	support
0	0.91	0.93	0.92	22684
1	0.93	0.91	0.92	22707
accuracy			0.92	45391
macro avg	0.92	0.92	0.92	45391
weighted avg	0.92	0.92	0.92	45391

Figure 2: Classification reports of SGD classifier and RNN

Reference list:

1. Goyal, A 2021, *Reddit dataset for Multi-task NLP*, kaggle, 21 June, viewed 12 June 2023, https://www.kaggle.com/datasets/amangoyl/reddit-dataset-for-multi-task-nlp?select=Dataset_Suicidal_Sentiment.csv.
2. Komati, N 2021, *Suicide and Depression Detection*, kaggle, 19 May, viewed 12 June 2023, https://www.kaggle.com/datasets/nikhileswarkomati/suicide-watch.
3. Yadhu, A 2022, *Predicting Suicide and Word Analysis*, kaggle, 2 July, viewed 4 July 2023, https://www.kaggle.com/code/yadhua/predicting-suicide-and-word-analysis.