Asian Journal of Electrical Sciences
ISSN: 2249-6297 (P); 2583-7923 (E)
Vol.12 No.1, 2023, pp.46-56
© The Research Publication, www.trp.org.in
DOI: https://doi.org/10.51983/ajes-2023.12.1.3694

The Accuracy Analysis of Different Machine Learning Classifiers for Detecting Suicidal Ideation and Content

Divya Dewangan[1], Smita Selot[2] and Sreejit Panicker[3]
[1]Research Scholar, [2]Professor and Head,
Shri Shankaracharya Technical Campus, Bhilai, Chhattisgarh, India,
[3]Training Head, Techment Technology, New STPI, Bhilai, Chhattisgarh, India
E-mail: divya93dewangan@gmail.com, smitaselot@sstc.ac.in, sreejit.bhilai@gmail.com

(Received 10 April 2023; Revised 29 April 2023; Accepted 21 May 2023; Available online 26 May 2023)

Abstract - **Suicide is the matter of purposely causing one's death and suicidal ideation refers to thoughts or preoccupations with ending one's own life. Studies have explored verbal and written communications related to suicide, including analyzing suicide notes, online discussions, and social media posts to identify linguistic and content markers that may help in early detection and intervention. The primary purpose of this study is to detect signs of risk of suicide/self-harm in social media users by investigating several frequency-based featuring and prediction-based featuring methods along with different baseline machine learning classifiers. The algorithms applied for analysis are Decision Tree, K-Nearest Neighbors, Random Forest, Multinomial Naïve Bayes, and SVM. Our experimental results showed that the best performance is obtained by the FastText embedding with SVM model having the highest accuracy of 93.76% which outperforms other baselines. The aim of this work is to learn the significance of analysis and do a comparative study of algorithms to find the best suited algorithm.**

Keywords: **Risk of Self-Harm/Suicide, Mental Health, Machine Learning Algorithms, Social Media, Frequency Based Featuring, Prediction Based Featuring**

I. INTRODUCTION

As per World Health Organization (WHO) 2022 data, close to 800,000 people a year making a suicide attempt and many more have serious thoughts of suicide. The rates of suicide can vary across different countries and regions, influenced by factors such as cultural, social, economic, and demographic factors. In many parts of the world, there are significant underreporting and challenges in collecting accurate data in suicide due to factors such as stigma, lack of awareness, and inadequate reporting systems [1].

Suicide is a complex and multifaceted phenomenon impacted by different types of factors such as mental health disorders, psychological, Access to lethal means, previous suicide attempts, hopelessness and depression, and substance abuse including alcohol, drug addiction, can impair judgment, increase impulsivity and contribute to a higher risk of suicidal behavior [2]. Social media platforms have increasingly recognized the need to address suicide prevention. Many platform have implemented policies, and features to provide resources, support, and interventions for users who may be at risk. Online communities and peer support groups can provide a sense of belonging, understanding, and connection for individuals who may feel isolated or stigmatized in their offline environments [3]. Researchers have explored the use of social media data in studying suicide-related behavior and trends. Analyzing posts, hashtags, and patterns of communication on platforms like twitter or Reddit can provide insights into risk factors, warning signs, and the impact of public health interventions [4].

The usage of Natural Language Processing (NLP) and Machine learning Techniques play a role in analyzing online communication and detecting patterns that may indicate a person's mental health, including suicidal thoughts or behaviors [5]. By leveraging these techniques, it becomes possible to identify specific linguistic markers, sentiment analysis, topic modeling, and other relevant features that may indicate distress or suicidal ideation in online communication. This can help in early detection intervention, and providing adequate assistance to those who may be in danger. It's essential to approach the topic of suicide with sensitivity and empathy and prioritize the well-being of individuals involved. It is important to seek help if someone is struggling with suicidal thoughts [6].

The objective of this study is to analyzing people's social media content by utilizing natural language processing (NLP) and by using machine learning methods, which make it possible to extract valuable insight that may indicate individuals at risk of self-harm or suicidal behavior. Thus, this work contributes to machine learning as well as the relevant research area. Studying mental health problems by proposing a generalized approach to detecting risk of self-harm that is especially effective when social media users are not cautious or are in denial.

The rest part of this paper is consolidated as follows. In Section II, we review previous related work on risk of self-harm. Section III describes the dataset, Different Feature extraction techniques and Methodology used in the research paper. Section IV describe experiment carried out on dataset, result and discussions and Section V, represents the conclusion.

II. RELATED WORK

Suicide is the biggest cause of death in which the life-time risk is 10-15 times higher than other mental issues. Increased mortality in patients with Mood disorders are almost twice as common in the general population. Suicidal threats, gestures, online posts and ideations are more common than complete suicide. However, suicide can occur as be accomplished contingently, all such hazards should to be taken seriously [2].

Chatterjee *et al.,* (2022) investigates the influence of personality traits of users having suicidal thoughts. Together, for this purpose, the authors also tried to cognize a subset of individual characteristics that may be indicators of suicide risk, using appropriate criteria. Extracted several features including linguistic, topic, sentiment feature, TF-IDF and temporal. They designed different machine learning classifiers, where the Logistic regression classifier outperformed the other classifiers [4].

The work of Birjali *et al.,* (2017) focused on the consideration of constructing a vocabulary associated with suicide and computing on an algorithm related with semantic analysis between tweets. The purpose of this paper is predicting suicidal thought and suicidal role using data collected from social Medias [7].

Burnap *et al.,* (2017) analyzed the twitter posts to extract the features such as lexical, structural, emotive and psychological features and design a set of fundamental classifiers and ensemble them using decision methods i.e., the Rotation Forest algorithm and a Maximum Probability voting classification. The main aim of this work is to distinguishing between worrying language and petty context to suicide, report and awareness raising about suicide [8].

Zang *et al.,* (2021) collected data from online including suicidal notes, last statement written by prison and neutral posts. They proposed a transformer learning model named TransformerRNN which is a combination of transformer encoder and BiLSTM. The model extracts two types of features i.e., the relevant information and secret features, used to cognize suicide notes and evaluates the model using baseline approaches and achieves 94.9% F1-score [9].

Tadesse *et al.,* (2019) examines the relationship between depressive thoughts and usage of language used by users in posts. They estimate the strength of N-grams, LDA and LIWC techniques as single and their combinations to show the predictive power with proposed classification approaches. They achieves the accuracy of 91% in depression identification task [10].

Kumar *et al.,* (2021) proposed a multiplicative attention-based bidirectional gated recurrent unit. Attention mechanism solved the issue of remembering longer input sequences and capturing local and global reference of the features. Furthermore, additive and multiplicative attention mechanism represent context vectors in similar complexity [11]. Lasari *et al.,* (2022) conducted a literature study on how social media platform use affects the identification and classification of suicidal thoughts. In addition, they summarize and designed the techniques to classify and automatically detect whether the user has any risk of thoughts or ideas of suicide [12].

Salehi *et al.,* (2023) analyzed and separated Persian textual content pertinent to Domestic violence against women and used machine learning to predict the threats. But the limitation is that each post are labeled manually, and the model is not sensitive to language due to its complexity and different dialects [13].

Zulfiker *et al.,* (2021) have studied socio-demographic and psychosocial factors that are responsible for depression. Three different feature selection methods were used to extract the most appropriate features and six different machine learning classifiers to detect whether a person is depressed [14].

The work of Lekkas *et al.,* (2021) focused on predictive modeling of acute suicidal ideation. They employed an ensemble machine learning model and used predictors to capture the language used and the activities that occurred in the social networking data [15].

Berkelmans *et al.,* (2023) identifies populations at extremely high risk of suicide based on risk factors using a novel machine learning heuristic. They identified social isolation and socioeconomic hardship as standard separators with three sub-population at extremely high risk of suicide [16].

The work of Caicedo *et al.,* (2022) focused on determining a semi-supervised method using bootstrapping technique. This method automatically detects and classifies text extracted from social media related to depression and suicide. The experiments are performed using five data collections and applied two machine learning classifiers: SVM with Bag-of-words features and RASA with default feature extraction system [17].

Rubani *et al.,* (2023) created a novel dataset that was annotated with the help of psychiatrists and psychologists' consultation and employing a hybrid mechanism of feature engineering to extract the most appropriate features. They evolved a methodology that helps to segregate posts into three classes: high risk, moderate risk, and no risk [18].

III. METHODOLOGY

This Section describe suicidal content detection model with following layers: Dataset Description, Data preprocessing, Feature extraction technique and different classifiers. Fig. 1 shows the model Architecture. This model takes preprocessed text as input and classification label as an output.

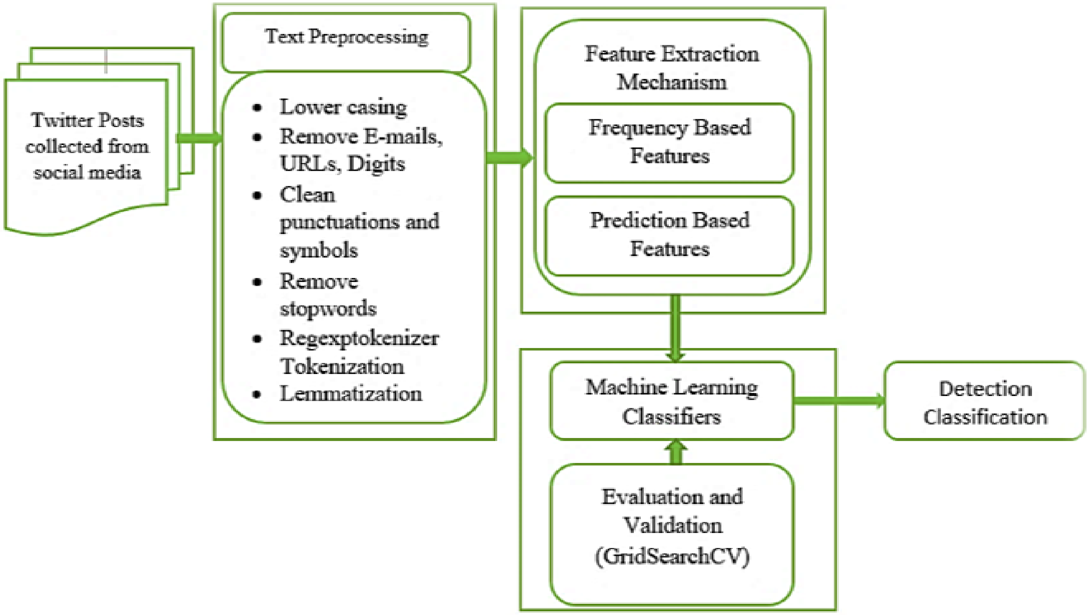

Twitter Posts collected from social media
Text Preprocessing
Lower casing
Remove E-mails, URLs, Digits
Clean punctuations and symbols
Remove stopwords
Regexptokenizer Tokenization
Lemmatization
Feature Extraction Mechanism
Frequency Based Features
Prediction Based Features
Machine Learning Classifiers
Evaluation and Validation (GridSearchCV)
Detection Classification

Fig. 1 Implementation Steps for Detecting the Suicidal Content

A. Dataset Description

In this work, we have used suicide detection dataset, which is comprised of 232,074 tweets, collected from kaggle website. A ratio of 70:15:15 has been adopted for splitting collected data into training, validation and test dataset. The dataset consists of 2 columns namely text and class. The text part contains the tweets, and the class part contains 2 categories: suicide and non-suicide as shown in Table I illustrated Suicide indicative distribution of dataset.

TABLE I SUICIDE INDICATIVE DISTRIBUTION OF DATASET

Dataset Information	**Quantity**
Total number of Post	232074
Suicide indicative post	116037
Non-Suicide indicative post	116037

B. Dataset Preprocessing

Data downloaded from online sources may contain several noises and due to their existence, it cannot be directly utilized for further process. The problem can increase further if there is redundant data, spelling mistake, emojis, URLs, and other unwanted characters in the data. Hence to ensure the computational model gets reliable predictive analysis, it needs to be preprocessed. The following preprocessing steps are performed on the dataset:

1. URL links, Emails, stopwords, punctuations, numbers and repeated characters are removed as a part of preprocessing.
2. All texts are converted into lowercase or normal form.
3. Using RegexpTokenizer to tokenize the data into tokens that matches alphanumeric characters. Here the regular expression '\w+' matches any sequences of one or more alphanumeric characters, so the tokenizer splits the text into tokens based on these matches.
4. Finally, it converts all tokens back to their original forms by applying Lemmatizer.

Here we are dividing the data into three sets such as training, validation and test sets. The training data set is used to train the model, the validation set is used to tune the hyperparameters of the model and prevents overfitting, and the test set is used to evaluate the model's performance.

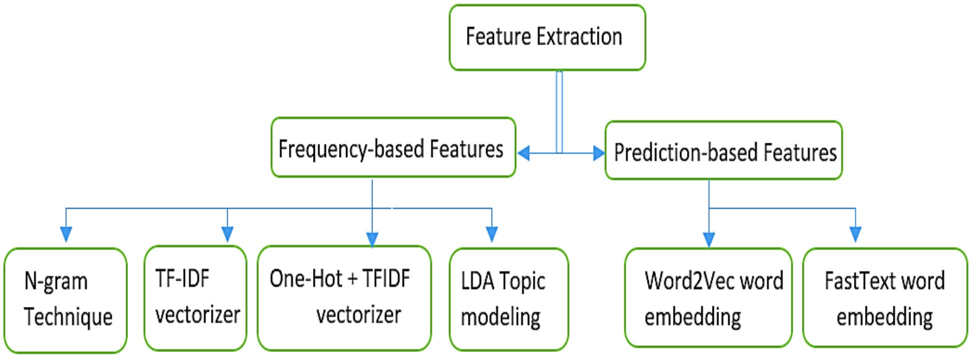


Fig. 2 Frequency-based and Prediction-based feature Extraction Techniques

C. Feature Extraction

Feature extraction is an essential step in collecting comprehensive knowledge about the users to provide high accuracy in detecting suicidal content. To demonstrate and describe suicide and non-suicide posts, we extract the different features in two broad categories namely, frequency-based and prediction-based featuring [19] as shown in Fig. 2.

Frequency-based features provide information about the prevalence or occurrence of specific words or phrases in a document or dataset. These features are often used in natural language processing (NLP) tasks, such as sentiment analysis [20] or text classification. Various frequency-based feature representations [21] are N-gram, TF-IDF, One-Hot Encoding, LDA (Latent Dirichlet Allocation) etc.

Prediction Based features are generated by a machine learning model itself. These features are learned by the model based on the patterns and relationships it identified in the input data. Once learned, the model can use these feature to make predictions on new, unseen data. Prediction-based feature are often used in application that involve high-dimensional input data, such as image, audio, or video processing. Prediction-based feature representation used are Word2Vec word embedding [20] and fastText embedding [6].

In general, frequency-based features are relatively simple to generate and can be effective in certain types of machine learning tasks such as text classification. On the other hand. Prediction-based features are more complex and require more computational resources to generate but can be highly effective in a wide range of machine learning applications that involve high dimensional input data [19]. Overall prediction-based features are an important tool in the machine learning toolkit and can be used to enhance the execution of a machine learning applications.

1. Frequency-based Features

a. N-gram Technique: To extract bigram features, the *CountVectorizer* from the Scikit tool Python module is employed. We initialized a *CountVectorizer* with *n-gram_range=(1,2)* to generate both unigrams and bigrams as features where each feature corresponds to a term or combination of adjacent terms in the text. Then we fit and transformed the vectorizer on the sentences. The resulting vector representation will be binary (0 or 1) indicating the presence or absence of each term in the text, and the dimensionality of the vector space will be limited to 1000.

The use of bigrams as features will capture some context information. While removing common stop words and infrequent terms can improve the quality of the resulting features representation.

b. Term Frequency: Inverse Document Frequency Vectorizer (TF-IDF): The *TFIDF Vectorizer from* the Scikit-learn Python module is utilized and is used to extract TF-IDF features. It converts a set of text corpus to a matrix of TF-IDF features. TF-IDF considers the frequency of words within a document and also the rarity of the words across all documents in order to weigh the emphasis of each word in the document [21].

The *TF-IDF Vectorizer* class implements this approach by first creating a dictionary of all infrequent or unique words in the text corpus and then transforming each document into a vector representation where the value of each dimension corresponds to the TF-IDF score of a particular word. The dimensionality of the vector space will be limited to '*n_features=1000*'. The resulting vector representations can then be used as input to machine learning algorithms. Removing common stop words can improve the quality of the resulting feature representation.

c. One-Hot + TFIDF Vectorizer: One-Hot encoding and TF-IDF vectorizer are two different techniques for representing text data in machine learning. One hot encoding represents each unique word in a text as a binary feature vector, where each dimension corresponds to a unique words in the vocabulary and has a value of either 0 or 1 depending on whether the word appears in text or not. TF-IDF vectorizer, on the other hand, assigns a weight to each word in a text based on its frequency in the text and its rarity across all texts in the corpus [21].

Here *Count Vectorizer* is used to perform the initial Vectorization of the text data, with the *binary=True* argument used to specify binary encoding that is nothing but One-Hot encoding. The resulting binary feature vector are then transformed using *TFIDF Transformer* to weight the features by TF-IDF. The benefits of combining One-Hot encoding with TF-IDF Vectorizer are

i. One-hot encoding can lead to a high-dimensional feature space, which can be computationally expensive and may lead to overfitting. By combining this with TFIDF, the number of features can be reduced while still capturing the important information in the text.
ii. One-hot encoding captures local information about the presence or absence of each features, while TF-IDF captures global information about the importance of each feature across all documents. By combining the two, both local and global information can be captured and used to better represent the text data [22].

d. LDA Topic Modeling: Latent Dirichlet Allocation (LDA) is a topic modeling algorithm that extracts features from text data by identifying patterns of co-occurring words in a corpus of documents. Here, each document is a combination of topics, and each topic is a probability distribution over words in the vocabulary [9]. During training, the LDA algorithm iteratively assigns words in each document to a topic and updates the topic distributions based on the words assignments in the corpus. The output of the LDA model is a set of topics, each represented as a probability distribution over the vocabulary. Where the most probable words for each topic can be considered as the features [4]. The number of specified topics determines the classification accuracy. Here we experimented with a variety of topic for study and found that 20 was an appropriate value. Below Table II shows some of the topics generated using LDA.

TABLE II TOPICS CONSTRUCTED USING THE LDA FEATURES ON THE SUICIDAL DATASET

Topic 1	filer, horny, text, notable, omg, cat, Netflix, fortnight, web, agent, lad, duck, fury, femboy
Topic 2	help, need, please, yawl, suggestion, clinic, cum, struggles, freedom, twitch, Texas, streaming, celebrate, suspicious, singing
Topic 3	Pop, trapped, cough, face, care, fake, bullet, stream, swallow, fly, mountain, hole, train.
Topic 4	wanna, talk, anyone, someone, bored, want, im, chat, damn, please, greate.
Topic 5	Kill, it, get, want, go, way, gun, head, eye, would, jump, one, dream, die, take.

2. Prediction-based Features

a. Word2Vec word embedding: Word2vec is a neural network-based technique for generating distributed representations of words in a corpus. These representations, also known as word embeddings, capture the semantic and syntactic interpretation of words and their context in a corpus [23].

Word2vec model does Vectorization of sentences by first generating vector representations for individual words in the sentence. To vectorize sentence using Word2vec [21], we typically rake the average of the word of all the words in the sentence. This results in a vector representation of the sentence that captures its overall meaning. This approach is effective because it preserves the relationships between words and their contexts in the corpus and allows us to represent sentences in a high-dimensional space where similar sentences are closer together [20].

To obtain a vector representation for the entire sentences, we simply take the average of the word vectors:

$$v(S) = \left(\frac{1}{n}\right) * (\ v(w1) + v(w2) + \cdots + v(wn)) \qquad (1)$$

Where a sentence *S* consisting of n words *{w1, w2... wn}*. A d-dimensional vector represented for each word in the sentences and the vector for word *wi* is denoted as *vi*. The vector *v(S)* represented the sentence *S* in a d-dimensional space, where similar sentences are expected to be closer together.

b. FastText Word Embedding: FastText word embeddings is an extension of the Word2Vec model that also takes into account the sub-word information. Instead of learning vector representations only for complete words, it learns vector representations for character n-grams, which are sub-words of a given length. To vectorize a sentence, we first represent each word in the sentence as the sum of its individual sub-word embeddings. The sub-word embeddings are then averaged to obtain a vector representation for the sentence [6].

To obtain a vector representation for each word, we use a bag-of-n-grams approach, where we represent each word as the summation of the embeddings of all its n-grams:

$$v(wi) = \sum vj \ \ (where\ j\ is\ an\ n-\ gram\ of\ the\ word\ wi) \ \ (2)$$

The vector representation for each word is then averaged to obtain a vector representation for the entire sentence:

$$v(S) = \left(\frac{1}{n}\right) * (\ v(w1) + v(w2) + \cdots + v(wn)) \qquad (3)$$

Here a sentence *S* consisting of n-words *{w1, w2... wn}*. Each sub-word in the sentence can be represented as a d-dimensional vector and the vector for sub-word is denoted as *sj as vj.* The vector *v(S)* represented the sentence *S* in a d-dimensional space, where similar sentences are expected to be closer together.

c. Min-max Normalization: The vector representation may contain negative values in features and this can be caused by the use of a signed or continuous bag-of-words approach to training the model. In this approach we are applying min-max normalization to the Wod2Vec features and FastText features, to transform them into a range between 0 and 1. This technique can help to transform the features into a more standard range without losing the negative values and ensures negative values scaled between 0 and 1.

D. Algorithm Used

The ascertaining suicide ideation posts on social media can be a classification problem of supervised learning. For a given dataset that consists of posts $X_{i=1,\ 2...n}$ represented as features and their corresponding labels $Y_{i=1...n}$, represented as suicidal or non-suicidal. The underlying function yi =F(Xi) is learned by training a supervised classification model that can accurately predict the label yi for new unseen post, where yi =1 and $y_i = 0$ indicates a suicidal text and non-suicidal text respectively [4]. There are five basic machine learning algorithms employed to classify the suicidal ideation and content.

1. Naïve Bayes Classifier: Naïve Bayes classifier is a probabilistic algorithm that follows Bayes' theorem to make predictions. This prediction is based on the probability of an event occurring given some evidence. It assumes that the features used to make predictions are independent of each other even though this may not be true in reality [24]. In the context of Naïve Bayes, to predict the class C for given some input features X, the mathematical notation is:

$$P(C|X) = P(X|C) * P(C)/P(X) \qquad (4)$$

Where P(C|X) represents the posterior probability for C given X. P(X|C) represents the likelihood of X given C. P(C) represents the posterior probability of C and P(X) represent the prior probability of X.

To classify a new input, the posterior probability are computed for each possible class and the class with the highest probability is chosen. Practically, the calculation can be simplified by assuming the features are independent given the class, so that can be written as follows:

$$P(X|C) = P(x1|C) * P(x2|C) * \dots * \ P(xn|C) \qquad (5)$$

Where x1, x2… xn are the individual features of X. we can estimate the likelihood and prior probabilities from the training data using maximum likelihood estimation or Bayesian estimation.

2. Decision Tree Classifier: Decision Tree Classifier is a machine learning classifier that uses a tree-like structure to make predictions or decisions based on a set of statements or rules [24]. Mathematically, a decision tree can be a form of a binary tree where each branch or internal node describes a decision based on a feature and the last level also known as the leaf node represents a class label. The tree is created by recursively separating the data into subsets on the basis of features values. The algorithm selects those feature at each that maximize the information gain. Information gain measures the entropy (or impurity) reduction of the data after the segmentation. The goal is to maximize the purity of the subsets so that each leaf node contains a homogeneous class.

The information gain can be computed using the formula:

$$IG(D,F) = H(D) - H(D|F) \qquad (6)$$

Where *IG(D, F)* is the information gain of features *F* on dataset *D, H(D)* is the entropy of the dataset *D* which measures the uncertainty of the class labels. *H(D|F)* is the weighted average of the entropy of the subsets after the split based on feature F.

The entropy can be computed using the formula:

$$H(D) = \ -\sum(p_i * log2(p_i) \qquad (7)$$

Where p_i is the ratio of instances in D that belong to class *i*.

3. Random Forest Classifier: The Random Forest classifier is a conjunction of several decision trees. It is an ensemble learning algorithm that improves prediction accuracy and reduces overfitting. [20]. At each split, the training data and features are randomly sampled, and the algorithm builds a set of decision trees from those samples. Then accumulates the prediction of the individual trees to make the final prediction. The idea is to reduce the variance of the model by creating diverse tress that capture different aspects of the data, while maintaining a low bias by combining their predictions [23].

The predictor of classifier is given as:

$$F(x) = argmax_i\{\sum_{b=1}^{B} T(A(B,\theta_b))\} \qquad (8)$$

Where the F(x) is the random forest model, B is the number of sub-trees, and θ_b characterizes the both random forest trees [4].

4. Support Vector Machine (SVM): Support Vector machine is a popular supervised machine learning algorithm that is especially adequate for solving problems of binary classification. The fundamental idea of SVM is to discover an optimum hyperplane that segregates the data points into different classes with the largest possible margin. The hyperplane is a decision boundary that maximizes the distance between the hyperplane and the support vectors (data points closest to the boundary) of each class [24]. By maximizing the margin, SVM aims to obtain good generalization performance and robustness to new data. For the binary classification problem, the hyperplane is defined by the below equation:

$$w\hat{\ }T \ * x + b = 0 \qquad (9)$$

Where w is a vector of weights that determines the hyperplane orientation, and b is a scalar that determines the hyperplane offset from the origin. Mathematically, given a set of training data {(x1, y1), (x2, y2),… (xn, yn)}, where xi is a vector of input features and yi is the corresponding class label. SVM solves the following optimization problem, to find the optimal hyperplane:

$$minimize\ \left(\frac{1}{2}\right) ||w||^2 + C * sum_i = 1^n \max\left(0, 1 - y_i(w^t x_i + b)\right)$$

$$subject\ to\ \ y_i(w^t x_i + b) \geq 1 \ \ for\ i = 1,2,3,\dots,n \qquad (10)$$

Where $||w||$ is the Euclidean norm of the weight vector, C is a regularization parameter with a trade-off between maximizing margin and minimizing classification error, and the constraints ensure that the data points are correctly classified with a margin of at least 1 [20].

5. K-Neighbor Classifier (KNN): K-Nearest Neighbor is a popular machine learning algorithm used for classification and regression problems. KNN is a type of instance-based learning, where the classifier stores all training data points and uses them to make predictions on new data points. In case of classification problem, the KNN algorithm works as follows:

a. Given a new input data point x, find the k-nearest neighbors in the training data on the basis of some distance metric.
b. Assign the class label of the new data point to the majority class label among its k-NN.

The performance of a KNN classifier can be affected by distance metric that measures the similarity between data points. The value of K can also affect the performance of the model, a small k-value may lead to overfitting and a high variance, while a large k-value may lead to underfitting and a high bias [24]. The optimum value of k depends on the specific problem and data and can be determined using techniques such as cross-validation or a separate validation dataset.

Tuning the hyperparameters of each classifiers for Frequency-based Features: When building a model, we need to assess the performance of a model on an independent dataset. The validation set has been utilized to tune the model's hyperparameters, and to prevent overfitting. We have defined the hyperparameters for each model separately, which we want to search over. We then create a GridSearchCV object and pass the model, the hyperparameters to search over, and the number of folds for cross-validation.

In this case, we passed cross-validation cv = 5. We then call the 'fit' method on the 'GridSearchCV' object to perform the hyperparameter search. After the investigation is completed, we print out the best hyperparameters and the corresponding score by accessing the 'best_estimator_' and 'best_score_' attributes of the 'GridSearchCV' object. The best hyperparameters for each model that we have been found using 'GridSearchCV' method shown on Table III.

TABLE III SHOWN THE BEST HYPERPARAMETERS FOR THE EACH CLASSIFIERS FOR FREQUENCY-BASED FEATURES

Name of Classifier	Defined Hyperparameters	best_estimator_		best_score_

KNN	param_grid = {
	'n_neighbors': [3, 5, 7, 9], 'weights': ['uniform', 'distance']
	}
		N-gram	{n_neighbors=3}	0.7405

		TFIDF	{'n_neighbors': 3, 'weights': 'uniform'}	0.5151
		One-Hot + TFIDF	{'n_neighbors': 5, 'weights': 'distance'}	0.7405
		LDA	{n_neighbors=9}	0.7250

Naïve Bayes	param_grid = {
	'alpha': [10**-3, 10**-2, 10**-1, 10**0, 10**1, 10**2, 10**3]
	}
		N-gram	alpha=10	0.8510

		TFIDF	alpha=1	0.8724
		One-Hot + TFIDF	alpha=0.1	0.9093
		LDA	alpha=0.1	0.5923

Decision Tree Classifier	param_grid = {
	'criterion': ['gini', 'entropy'], 'max_depth': [3,5,10, None], 'min_samples_split': [2,5,10], 'max_features': ['sqrt', 'log2', None]
	}
		N-gram	{ max_depth=10, min_samples_split = 10}	0.8431

		TFIDF	{max_depth = 10, random_state = 42}	0.8434
		One-Hot + TFIDF	{'criterion'= 'gini', 'max_depth'= 10, 'max_features'=None, 'min_samples_split'= 10}	0.8425
		LDA	{ criterion='entropy', max_depth=10, max_features='log2', min_samples_split=5 }	0.7630

Random Forest Classifier	param_grid = {
	'n_estimators': [100, 200], 'max_depth': [None], 'max_features': ['sqrt', 'log2']
	}
		N-gram	{ max_features='log2', n_estimators=200 }	0.8781

		TFIDF	{ max_features='log2', n_estimators=200 }	0.8861
		One-Hot + TFIDF	{ max_features='log2', n_estimators=200 }	0.8877
		LDA	{ max_features'= 'sqrt', 'n_estimators'= 200 }	0.7580

SVM	param_grid = {
	'alpha': [0.001,0.01,0.1,1], 'penalty': ['l1', 'l2', 'elasticnet'], 'max_iter': [200, 500],
	'tol': [1e-3,1e-4],
	'random_state': [42]
	}
		N-gram	{alpha=0.001, max_iter =200, penalty = 'elasticnet', random_state=42,tol = 0.0001}	0.8926

		TFIDF	{alpha=0.1, max_iter=200, random_state=42, tol=0.0001}	0.9044
		One-Hot + TFIDF	{alpha=0.001, max_iter = 200 , random_state=42}	0.8979
		LDA	{alpha=0.01, max_iter = 200, random_state = 42}	0.7392

IV. EXPERIMENTAL ANALYSIS

This Section summarizes evaluation metrics, results, and findings of this project. This study aims to determine whether a user is trying to attempt suicide by investigating their post and to do a comparative analysis of algorithms to find the best suited algorithm. This research ponders 70% of the dataset for training and remaining 15% for validation and 15% for testing purposes. Python 3.10.7 64bit on a Jupyter Notebook 6.4.12 has been utilized in this experiments. Algorithm Selection and hyperparameter tuning help the model perform optimally for a particular dataset and prevent overfitting.

Evaluation Metrics: Evaluation metrics are used to estimate the performance of the model. Four evaluation metrics have been used for performance appraisal in research - precision (P), recall (R), weighted average F1-score and accuracy (A).

Accuracy (A): Accuracy (A) is the proportion of correctly classified instances.

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

Precision (P): Precision is the proportion of TP among all positive predictions.

$$P = \frac{TP}{(TP + FP)}$$

Recall (R): Recall is the proportion of TP among all actual positive instances in the data.

$$R = \frac{TP}{(TP + FN)}$$

Weighted Average F1-Score (F): F1-score is the harmonic mean of precision and recall, and it balances between the two metrics.

$$\mathrm{F} = 2 * \frac{(\mathrm{P} * \mathrm{R})}{(\mathrm{P} + \mathrm{R})}$$

Result and Findings: Five necessary classifiers are implemented to depict the importance of variant features compiled from the dataset. Basic feature extraction techniques such as N-gram, TFIDF, One-Hot + TFIDF, LDA, and Word embeddings such as word2vec and FastText have been utilized to generate word vectors with 100 dimensions from the training data.

Moreover, 'GridSearchCV' technique is used to optimize hyperparameters and reduce the risk of overfitting in machine learning models. The analysis result are shown in Table IV that shows the performance of five implemented model with the different features. Here the dataset are balanced so the accuracy and F1-score value both are equivalent. The Accuracy graph are shown in Fig. 3.

TABLE IV PERFORMANCE RESULTS OF EACH CLASSIFICATION MODELS

Classifiers Name	Feature Extraction Techniques		Accuracy	Precision	Recall	F1-score
Naïve Bayes	Frequency-based Features	N-gram	85.28	79.63	94.96	85.28
		TFIDF	87.27	83.53	92.95	87.27
		One-Hot + TFIDF	91.41	79.60	96.27	91.41
		LDA	69.71	68.94	72.15	69.71
	Prediction-based Features	Word2Vec embedding	86.44	82.59	92.31	86.44
		FastText Embedding	88.3	83.19	95.96	88.3
Decision Tree Classifier	Frequency-based Features	N-gram	84.32	87.73	79.95	84.32
		TFIDF	84.55	88.92	79.07	84.55
		One-Hot + TFIDF	84.31	87.46	80.25	84.31
		LDA	73.36	75.12	70.18	73.36
	Prediction-based Features	Word2Vec embedding	84.47	84.34	84.61	84.47
		FastText Embedding	85.94	84.34	84.61	85.94
Random Forest Classifier	Frequency-based Features	N-gram	88.95	87.7	90.7	88.95
		TFIDF	89.37	88.99	89.96	89.38
		One-Hot + TFIDF	88.44	89.32	87.41	88.44
		LDA	73.84	80.06	63.77	73.84
	Prediction-based Features	Word2Vec embedding	91.19	91.67	90.59	91.19
		FastText Embedding	91.97	92.5	91.33	91.97
Support Vector Machine	Frequency-based Features	N-gram	90.06	93.73	85.94	90.06
		TFIDF	90.97	92.44	89.3	90.97
		One-Hot + TFIDF	89.95	92.51	87.01	89.95
		LDA	68.55	84.51	45.72	68.55
	Prediction-based Features	Word2Vec embedding	93.01	92.95	93.06	93.01
		FastText Embedding	93.76	93.6	93.94	93.76
KNN Classifier	Frequency-based Features	N-gram	75.75	89.01	58.97	75.75
		TFIDF	53.09	75.51	9.7	53.1
		One-Hot + TFIDF	50.24	72.73	1.3	50.24
		LDA	74.63	73.6	77.13	74.64
	Prediction-based Features	Word2Vec embedding	87.12	89.0	96.52	87.12
		FastText Embedding	88.15	89.0	58.97	88.15

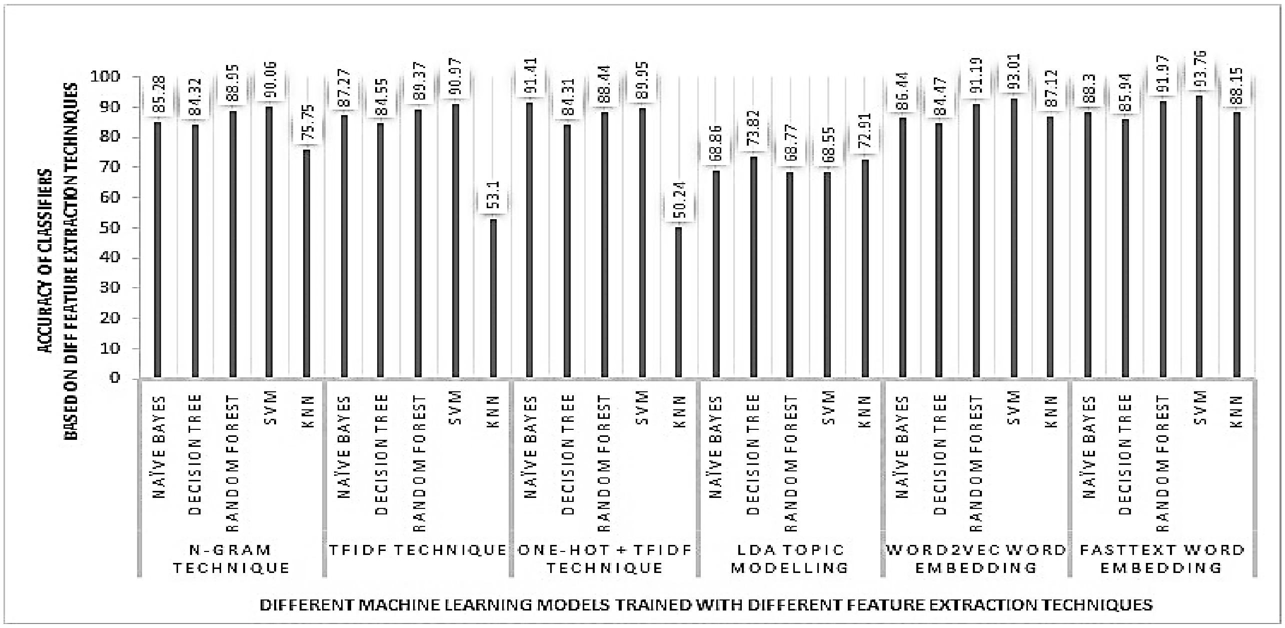
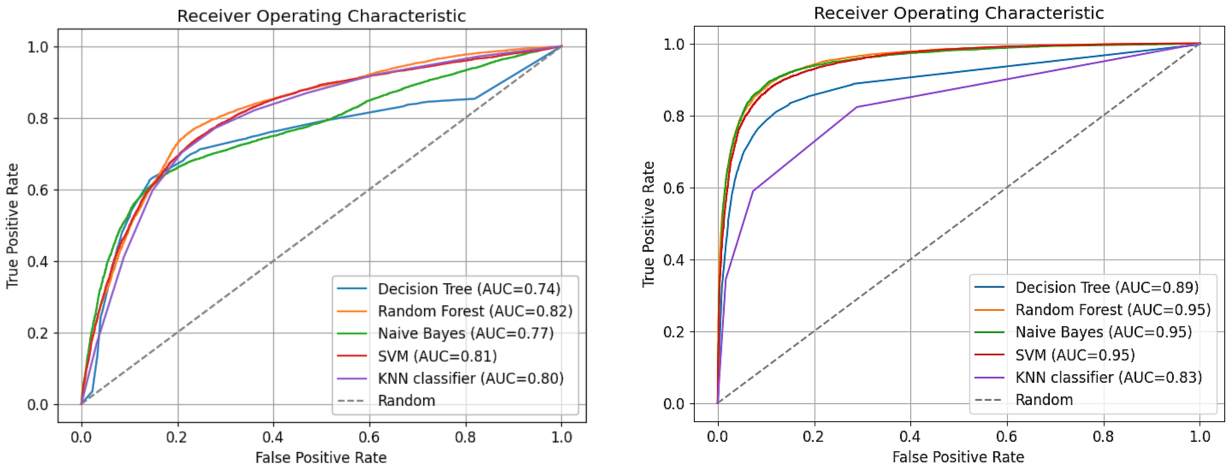


Fig. 3 Accuracy of Different Machine Learning Models Trained with Different Feature Extraction Technique

By examining the graph in Fig. 3. we can say that models performed well with word embedding feature vectors but the basic techniques like N-gram and TFIDF also producing better results too. The LDA features with all other models has achieves a lower accuracy score than other combinations. The FastText embedding with SVM model has highest accuracy of 93.76% among others. The accuracy of Word2Vec embeddings with SVM [23] is similar to FastText with SVM model of 93.01% accuracy. With 91% accuracy of FastText embedding with Random Forest classifier and One-Hot +TFIDF features with Naïve Bayes model come in second. Other combinations of feature vectors and models are giving approx. 85-90% accuracy that we can see in Fig. 3.

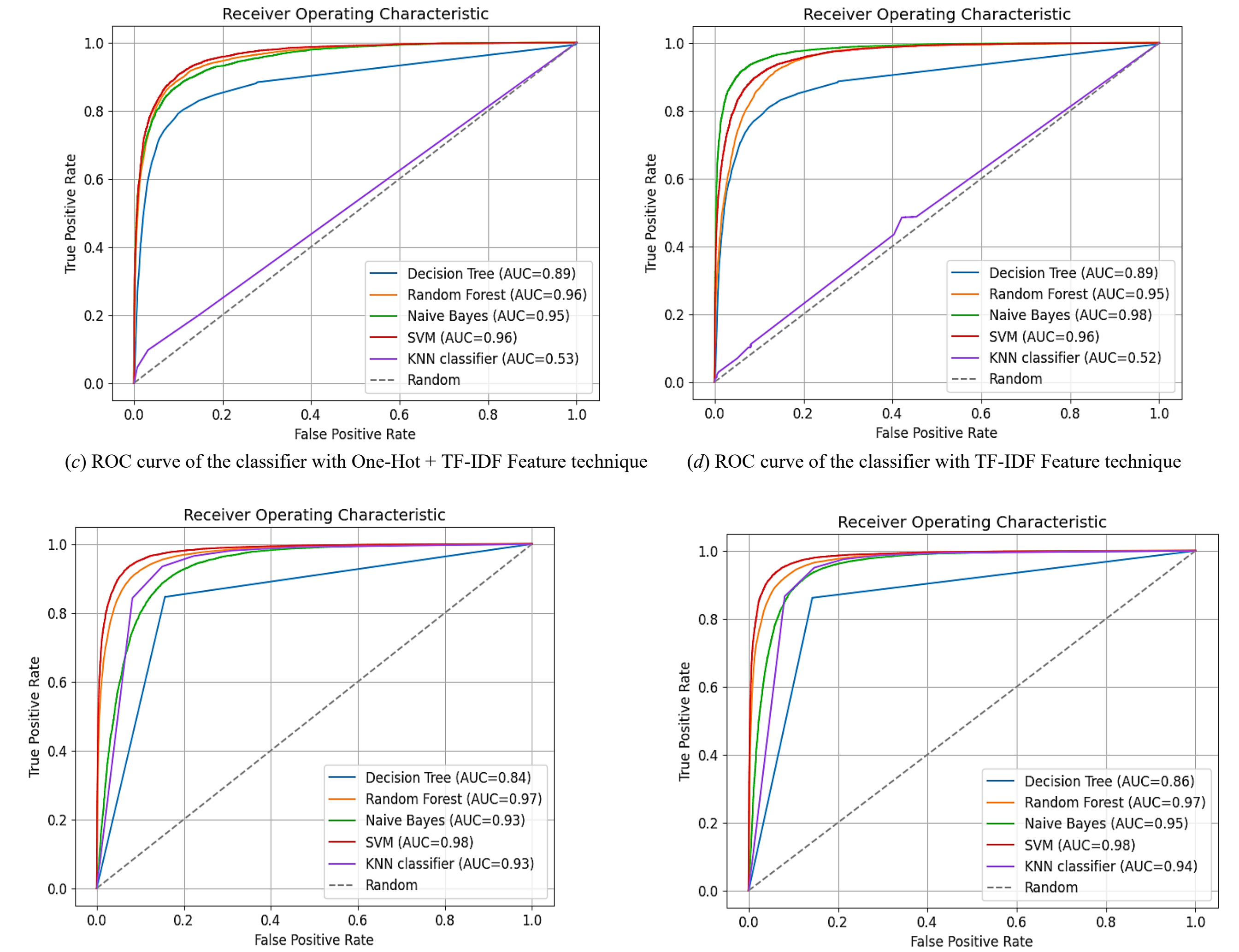
We also plotted the different AUC-ROC curve examine the execution of the models based of feature techniques used in Fig. 2. The AUC-ROC curve (Area Under the Receiver Operating Characteristic Curve) is a helpful mechanism for performance analysis of multiple classifiers in a binary classification task. By plotting the AUC-ROC curve for each classifier on the same graph, we can visually compare the performance of each model. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) on the y-axis and the x-axis respectively. Each point on the curve represents a different threshold setting for the model, which determines how it classifies instance as positive or negative [14]. The trade-off between sensitivity and specificity, represented by a point on the ROC curve close to the upper-left corner of the graph. Fig. 4 (a) - (e) shown the ROC curves of the used classifier using the different feature extraction techniques and word embeddings.

Comparing the result of the different graphs, it can be concluded that the SVM model has transcended the other model in terms of accuracy, AUC value, ROC curves, and other measured performance metrics. SVM model with FastText has shown the highest accuracy of 93.76 and has also obtained the highest AUC value of 0.98.



(*a*) ROC curve of the classifier with LDA Feature technique

(*b*) ROC curve of the classifier with N-gram Feature technique



(*c*) ROC curve of the classifier with One-Hot + TF-IDF Feature technique

(*d*) ROC curve of the classifier with TF-IDF Feature technique

(e) ROC Curve of the Classifier with Word2Vec Word Embedding Technique

(f) ROC Curve of the Classifier with Fast Text Word Embedding Technique

Fig. 4 ROC curve of the classifier for different Feature extraction and word embedding techniques

V. CONCLUSION

WHO recognizes that suicide prevention efforts must be integrated into the broader health system and policies, and involve collaboration across sectors such as education, social welfare, and justice. However, machine learning can provide insights and predictions, so it is crucial to integrate the findings with clinical expertise. Collaborating with mental health professionals is necessary to interpret the results, provide appropriate interventions, and ensure the safety and well-being of individuals at risk. In this article, we presented an analysis approach among different feature extraction techniques and different machine learning techniques for detecting the suicidal content and ideation on social media posts. In our study, FastText Embedding features work effectively with all other models and performing well than other used features. Word2Vec embeddings is performing as similar as FastText. But the main difference between them is that FastText considers the sub-word information in order to better capture the meaning of words and handle out-of vocabulary words while Word2Vec does not. This can lead to difference in the resulting sentence embedding and their ability to capture semantic similarity between sentences. The result can be improved by integrating questionnaires in consultation with mental health experts, it enhance the accuracy and validity of machine learning models for predicting self-harm risk. In the future work, the research can be extended by affined depressive posts, stress and anxiety based features, and negative sentiments. This would enhance the precision and accuracy of sentiment analysis for suicide content detection.

REFERENCES

[1] WHO Website, 2022. [Online]. Available: https://www.who.int/campaigns/world-suicide-prevention-day/2022.

[2] N. Ahuja, *A Short Textbook of Psychiatry,* 7th ed., Jaypee Brothers Medical Publishers, New Delhi, India, 2011

[3] J. Glazzerd, and S. Stone, Webpage on Selected Topics in child and Adolescent Mental Health, 2020. [Online]. Available: https://books.google.com.ng/books?hl=en&lr=&id=T3L8DwAAQBAJ&oi=fnd&pg=PA7&dq=restricting++social+media+use+in+mental+health&ots=b3Eq_VFXw&sig=8G92Jii84EYOB87UDtbue935NE4&redir_esc=y#v=onepage&q=restricting%20%20social%20media%20use%20in%20mental%20health&f=false.

[4] M. Chatterjee, P. Kumar, P. Samanta and D. Sarkar, "Suicide ideation detection from online social media: A multi-modal feature based technique," *International Journal of Information Management Data Insights,* Vol. 2, No. 2, Nov. 2022. DOI: https://doi.org/10.1016/j.jjimei.2022.100103.

[5] M. R. Islam, M. A. Kabir, A. Ahmed, A. R. M. Kamal, H. Wand and A. Ulhaq, "Depression detection from social media networks data using machine learning techniques", Health Information Science and System, Vol. 6, No. 8, 2018, DOI: https://doi.org/10.1007/s13755-018-0046-0.

[6] S. Ghosal and A. Jain, "Depression and Suicide Risk Detection on Social Media using fastText embedding and XGBoost classifier," in *International conference on Machine Learning and Data Engineering. Procedia Computer Science,* 2023, Vol. 218, pp. 1631-1639, DOI: 10.1016/j.procs.2023.01.141.

[7] M. Birjali, A. Beni-Hssane and M. Erritali, " Machine Learning and Semantic sentiment Analysis based Algorithm for Suicide Sentiment Prediction in Social Network," in *Proc. of the 8th International Conference on Emerging Ubiquitous System and Pervasive Networks,* pp. 1877-0509, 2017, DOI: 10.1016/j.procs.2017.08.290.

[8] P. Burnap, G. Colombo, R. Amery, A. Hodorog and J. Scourfield, "Multi-class machine classification of suicide-related communication on Twitter," *Online Social Networks and Media,* Vol. 2, pp. 32-44, 2017, DOI: https://doi.org/10.1016/j.osnem.2017.08.001.

[9] T. Zang, A. M. Schoene and S. Ananaidou, "Automatic identification of suicide notes with a transformer-based deep learning model," *Internet Intervention,* Vol. 25, 2021, DOI: https://doi.org/10.1016/j.invent.2021.100422.

[10] M. M. Tadesse, H. Lin, B. Xu and L. Yang, "Detection of Depression-Related Posts in Reddit Social Media Forum," *IEEE Access,* Vol. 7, pp. 44883-44893, DOI: 10.1109/ACCESS.2019.2909180.

[11] A. Kumar, T. E. Trueman and A. K. Abinesh, "Suicidal risk identification in social media," in *Proceeding of the 5th Internaltional Conference on AI in Computational Linguistics,* Vol. 189, 2021. DOI: 10.1016/j.procs.2021.05.106

[12] S. Lasri, E. H. Nfaoui and F. E. Haoussi, "Suicide Ideation Detection on Social Networks: Short Literature Review," in Proceeding International Conference on Innovative Data Communication Technology and Application, Vol. 215, 2022, DOI: 10.1016/j.procs.2022.12.073.

[13] M. Salehi, S. Ghahari, M. Hosseinzadeh and L. Ghalichi, "Domestic violence risk prediction in Iran using a machine learning approach by analyzing Persian Textual content in social media," *Heliyon,* Vol. 9, 2023, DOI: https://doi.org/10.1016/j.heliyon.2023.e15667.

[14] M. S. Zulfiker, N. Kabir, A. A. Biswas, T. Nazneen and M. S. Uddin, "In-depth analysis of machine learning approaches to predict depression," *Current Research in Behavioral Sciences,* Vol. 2, 2021, DOI: https://doi.org/10.1016/j.crbeha.2021.100044.

[15] D. Lekkas, R. J. Klein and N. C. Jacobson, "Predicting acute suicidal ideation on instagram using ensemble machine learning models," Internet Interventions, Vol. 25, pp. 100424, 2021, DOI: https://doi.org/10.1016/j.invent.2021.100424.

[16] G. Berkelmans, L. Schweren, S. Bhulai, R. V. D. Mei and R. Gillissen, "Identifying populations at ultra-high risk of suicide using a novel machine learning method," *Comprehensive Psychiatry,* Vol. 123, pp. 152380, 2023, DOI: https://doi.org/10.1016/j.comppsych.2023.152380.

[17] R. W. A. Caicedo, J. M. G. Soriano, and H. A. M. Sasieta, "Bootstrapping semi-supervised annotation method for potential suicidal messages," *Internet Interventions,* Vol. 28, pp. 100519, 2022, DOI: https://doi.org/10.1016/j.invent.2022.100519.

[18] S. T. Rabani, A. M. U. D. Khanday, Q. R. Khan, U. A. Hajam, A. S. Imran and Z. Kastrati, "Detecting suicidality on social media: Machine Learning at rescue," *Egyptian Informatics Journal,* Vol. 24, pp. 291-302, 2023 DOI: https://doi.org/10.1016/j.eij.2023.04.003.

[19] D. R. Kabul and A. V. Nimkar, "A survey on word embedding techniques and semantic similarity for paraphrase identification," *International Journal of Computational Systems Engineering,* Vol. 5, No. 1, 2019.

[20] S. Selot, and S. Panicker, "Comparative performance of Random Forest and Support vector Machine on Sentiment Analysis of Reviews of Indian Tourism," *IT in Industry,* Vol. 9, No. 2, 2021.

[21] Z. Keita, *Towards Data Science webpage on Text data representation with one-hot encoding,* Tf-Idf, Count Vectors, Co-occurrence Vectors and Word2Vec, 2021. [Online]. Available: https://towardsdatascience.com/text-data-representation-with-one-hot-encoding-tf-idf-count-vectors-co-occurrence-vectors-and-f1bccbd98bef.

[22] A. Chen, *Analytics Vidhya webpage on How does MeaLeon use NLP? Part 3: Some Results Comparing One Hot Encoding and TF-IDF,* 2020. [Online]. Available: https://medium.com/analytics-vidhya/how-does-mealeon-use-nlp-part-3-some-results-comparing-one-hot-encoding-and-tf-idf-e664c879882d.

[23] M. M. Trusca, "Efficiency of SVM classifier with Word2Vec and Doc2Vec models," in *Proc. of the 13th International Conference on Applied Statistics,* pp. 496-503, 2019, DOI: 10.2478/icas-2019-0043.

[24] D. Dewangan, S. Selot and S. Panicker, "Implementation of Machine Learning Techniques for Depression in Text Messages: A Survey," *i-Manager's Journal on Computer Science,* Vol. 9, No. 4, 2022, DOI: 10.26634/jcom.9.4.18549.