Evaluation of Cyberbullying using Optimized Multi-Stage ML Framework and NLP

Lida Ketsbaia, Biju Issac and Xiaomin Chen
Computer and Information Sciences, Northumbria University, UK

Abstract ~ Due to the evolution of technology, online hate is increasing, more specifically in areas of social media amongst the general population. Online hate has become a phenomenon that destructively impacts individuals, with victims suffering long-lasting mental and psychological issues. Since cyberhate is conveyed as an ever-growing social problem, researchers have tried to tackle the matter. One of the main methods researchers have focused on is through the means of Machine Learning to help classify whether a piece of textual data can be identified as cyberbullying or not. Therefore, the purpose of the research is to employ a multi-stage optimized Machine Learning Framework that will look at using a combination of two data balancing methods (RUS and SMOTE), the feature selection method PCA as well as the bio-inspired metaheuristic optimization techniques PSO and GA. The framework applied increases the performance of the Machine Learning Classifier Logistic Regression to help detect instances of cyberbullying. Furthermore, the paper will show the potential of using various NLP methods such as RoBERTa, XLNet and DistilBERT to find the most suitable model to use within the textual analysis of cyberhate.

I. INTRODUCTION

Bullying has increasingly been acknowledged as a serious concern amongst individuals, more specifically adolescents. It is identified as the notion of exposing "victims" to severe negative actions over a prolonged period for which there is an imbalance of power. Bullying takes on various forms, mostly physical, verbal, and relational [1]. However, due to the advancements of technology and social media platforms, bullying has evolved into cyberbullying, where the "relational" and "verbal" forms of bullying can occur [1].

Technology plays an integral role in regard to communication in today's society. This is due to individuals having more distinct ways to access the Web, thus enabling people to go online more frequently. The internet is used as a beneficial way for people to interact with each other thus facilitating a new method of human interaction. However, it has caused an increased number of problems due to the freedom and anonymity it allows. Therefore, as technology evolved, bullying has flourished.

Cyberbullying has been compared to traditional bullying in three ways. Firstly, from the aggression portrayed, since the "Bully's" intention is to harm an individual [2]. Regardless of cyber-bullying not taking the form of physical aggression, it is still a form of aggression expressed using technological mediums. The various emotions experienced by victims include embarrassment, depression, anxiety, as well as a low self-perception. The second comparison is based on the notion that cyber-bullying is repetitive [2]. Social media messages, posts, emails, comments, and pictures can often be sent out multiple times, further harassing an individual. Lastly, Cyberbullying links to traditional bullying since an imbalance of power occurs [2].

Despite the similarities between the two problems, there are ways for which the two differ. Cyberbullying enables the option of anonymity by using false identities. The unawareness of the perpetrator may cause more psychological trauma to victims. Additionally, due to anonymity, "cyber-bullies" may go to much further lengths to torment and taunt their victims. People are more likely to partake in such activities if their identity is not known. Furthermore, cyber-bullying can occur from the comfort of a victim's home producing the sentiment that they are never safe.

Based on further research being conducted on the matter, certain statistics were identified [3]:

1. Victims of cyberbullying are 1.9 times more likely to commit suicide.
2. Only 35% of UK students have never been a victim of cyberbullying.
3. 68% of adolescents that have gone through online harassment have experienced mental health issues.
4. 60% of adolescents have witnessed someone going through harassment on social media.
5. 71% of survey participants do not feel like social platforms are doing enough to fight against the issue of online abuse.
6. 36.5% of individuals that are aged between 12 and 17 have had a bully target them at least once in their lifetime.
7. 42% of LGBT youth have experienced cyberbullying. 35% received online threats, 58% have been a victim of hate speech at least once.

Such statistics detect the need for a solution. Individuals are psychologically and mentally impacted by their victimization and are increasingly more likely to commit suicide; this notion alone should be enough for the problem to be resolved. Despite the continuous advances in cyberbullying research, there are still areas that require improvement. Therefore, this research proposes a Multi-stage Machine Learning (ML) optimized framework that increases the performance of Logistic Regression to detect cyberbullying behaviour. Additionally, the paper will look into DistilBERT, RoBERTa, and XLNet to analyze their performance on online hate datasets.

II. METHODOLOGY

As previously mentioned, this research proposes a framework as well as a transformer-based Machine Learning approach. Numerous techniques are employed at different stages to help with this implementation.

A. Dataset

Firstly, the research employs two publicly available datasets. All datasets acquired came in csv form with a text column and a corresponding label column identifying whether the text is identified as “hate" or non-hate":

Dataset 1 is based on the article “Automated Hate Speech Detection and the Problem of Offensive Language” [4]. The dataset was created using Twitter’s API. Tweets were searched containing terms from a lexicon resulting in tweets from 33,458 Twitter users. A random sample of 25,000 tweets was chosen and labelled as one of three categories: “hate speech”, “offensive”, “neither hate-speech or offensive” [4].

The resulting sample consisted of 24,802 labelled tweets. Only 5% of the tweets were coded as hate speech, 76% were identified as offensive language, and the remainder were non-hate/offensive. Due to the research specifically targeting hate speech, this dataset only used the “hate speech” and “non-hate speech” data.

Dataset 2: The researchers created a dataset regarding cyberbullying due to the lack of cyber-bullying datasets within research. The text in the dataset were marked as “cyberbullying” based on whether two of the annotators identified the text and “cyberbullying”. The resulting dataset contains 12,772 samples, with 86% being labelled as “non-cyber bullying [5].

B. Data Pre-Processing.

Furthermore, as the research centres on textual data, pre-processing is increasingly essential. Due to social media platforms being the main area where the data is extracted, the text is largely unstructured. The pre-processing techniques implemented facilitated in a cleaner dataset where noisy data is removed, allowing the performance of the ML framework as well as the transformer-based Machine Learning approach to increase. Converting text into something that can be understood by the models is an intricate process and has three distinct parts. Firstly “Cleaning” was implemented, which removes parts of text that do not hold any value to the data. Normalization was then used to transform the text into a canonical form and, lastly, stemming, which reduces the inflexion in words.

C. Text to Numeric Data representation

In order to encode text into numeric data, the creation of pre-trained algorithms such as Term Frequency Inverse Document Frequency and Count Vectorizer is required. The modules “CountVectorizer” and “TfidfTransformer” from Scikit-learn are used to assign numbers to each word and provide its occurrence within a text.

Count Vectorizer is where vocabulary is generated. It is a simple yet efficient way of word vectorization. The basic idea for a count vectorizer is to create a vector that has the same dimension as the size of a sentence. It uses the bag of word approach and ignores the text structures, and only extracts information from the word counts. Sklearn enables Count vectorizer to be utilized whilst excluding English stopwords, which is text such as: the, is at, etc.

TfidfTransformer was then applied to count the frequencies of each word within the text document. Term Frequency signifies the word counts of all words in the input document, and Inverse Document Frequencies are computed for each word in the input in which the most frequently occurring words will be assigned a lower score, and the least occurring words will be assigned a higher score. TfidfTransformer returns a normalized vector where the highest score is 0, and lowest score is 1.

D. Imbalanced Data

To handle the issue of imbalanced datasets, the research uses a combination of two balancing techniques: SMOTE and Random under Sampling (RUS).

1) SMOTE

SMOTE (Synthetic Minority Oversampling Technique) is a minority class oversampling technique whose aim is to create synthetic instances of a minority class, thus helping reduce the class imbalance that largely impacts the performance of models.

2) Random Under Sampling

The random under-sampling method removes examples from the majority class to make the dataset more balanced. Such a method tries to balance the distribution of classes by randomly removing majority class samples. However, the obstacle with using under-sampling methods is that it can dispose of useful information.

The reason for using a combination of the two balancing techniques was to ensure that not many synthetic samples were produced whilst still balancing the data using RUS.

E. Feature Selection

Once incorporating the balancing measures, feature selection was applied. Features that are irrelevant or redundant can lead to machine learning classifiers to not performing as well; additionally, it can lead to classifiers converging at a much

slower rate. Therefore, PCA (Principal component analysis) technique is incorporated through Sklearn. PCA is used for "dimensionality reduction, which involves zeroing out one or more of the weakest principal components, resulting in a lower-dimensional projection of the raw feature data that preserves the maximal data variance. The dimensionality reduction process is achieved through an orthogonal, linear projection operation" [6]. Moreover, PCA is a machine learning algorithm that finds important variables that can be beneficial for further classification tasks.

The PCA is given using the following steps [7]:

1. $X \rightarrow N\ x\ d$ Matrix is created, with a one-row vector x^n in X
2. $X \rightarrow$ Subtract mean x from each row vector x^n in X
3. $\sum \rightarrow$ convergence matrix of X
4. Find the eigenvectors and eigenvalue of $\sum$
5. PC's $\rightarrow$ the M eigenvectors with largest eigenvalues

F. ***Optimization***

During the multi-stage machine learning optimization framework, the last implementation that took place was Particle Swarm Optimization and Genetic Algorithm to optimize the results of Logistic Regression.

1) ***Logistic Regression***

Logistic Regression is the advancement of linear regression techniques for circumstances where outputs are categorical variables. LR has largely been applied in areas of data mining and machine learning problems and has produced substantial results hence the reason it is used in the research. Logistic Regressions is acknowledged as a statistical method for analysing datasets where there are one or more independent variables that establish an outcome. The Classifier belongs to the family recognized as the exponential or log-linear classifiers. Log-linear classifier operates by extracting a set of weighted features from the input and combining them linearly. Logistic regression classifies an observation into one of two classes, where the output is a binary representation i.e 1 (Hate, Bullying, True etc.) or 0 (NON-HATE, FALS etc.). The objective is to find the best fitting model to distinguish the relationship between the dependent variables and a set of independent variables. Moreover, Ghosh and Sanyal described that "the probability of a feature vector i existing with positive class can be measured by logistic regression as given in Equation 1[8]:

$$P(c = 1|\ f) = l(f) = \frac{1}{1+ e^{wTf}} \qquad (1)$$

where P(c=1|f) refers to the probability of document `f' of class `c'. `w' indicates the feature-weight parameters to be estimated"[8].

2) ***Particle Swarm Optimisation***

Particle Swarm Optimisation (PSO) is a meta-heuristic optimization technique that mimics the notion of a flock of birds. PSO has been described as a simple optimization; however, regardless of it being a simple approach, it has been found to converge to the optimum within a wide variety of optimization problems. The underlying concept of PSO is that information is optimized not only by personal experience but from social interaction in the population [9].

All particles found in PSO is a point in the d-dimensional search space. "Particles have their own memories that record their best experience in the search space along with the best experience encountered by the swarm to find the best solution. The main purpose of PSO is to generate an optimal solution based on personal experience as well as information being shared amongst individuals within a swarm. Each individual solution in PSO moves in the search space with a dynamically adjusted velocity that is affected by its own experience and the global experience of other particles [9]".

PSO was initially produced to solve problems in continuous-numbers search space, yet, feature selection, as in many other optimization problems, occur in discrete search spaces. Therefore, Kennedy and Eberhart [9] established a binary version of the PSO to confront the optimization problem found within discrete domains; the binary version is identified as BPSO. While using BPSO the position of the particles are expressed in two terms: 1 or 0 (or on and off). If there is a particle x on d-dimensions, then its position can be defined as[10]:

$$x = [\ x_1 x_2 x_3 \ldots x_d]\ where\ x_i\ \in 0 \qquad (2)$$

Once BPSO is implemented, the best position is acquired, the binary array can be interpreted simply as turning a feature on and off. The features that are turned on will be perceived as the selected features. The classifiers can then be trained using only these features while dropping the others.

The objective function is described as [10]:

$$f(X) = \alpha(1 - P) + (1 - \alpha)\left(1 - \frac{N_f}{N_t}\right) \qquad (3)$$

The "α hyperparameter decided the trade-off between the classifier performance P, and the size of the feature subset N_f with respect to the total number of features N_t"[10].

The implementation of PSO can be viewed in Figure 1.

Figure 1: Particle Swam Optimisation (PSO) Implementation

ALGORITHM 1: PSO ALGORITHM

```
t ← training samples of textual data after pre-processing
T ← testing sample
Y ← labels corresponding to training data.
y ← labels corresponding to testing data.
Def Objective function:
    Computes the Objective function per particle
    Alpha: weight for trading off classifier performance and number of
    features
    M: binary mask obtained from binary PSO, used to mask features
    if (m) == 0:
            X_subset = t
    else:
            X_subset = t [:,m==1]
    Perform classification and store performance in P
    Compute for the objective function
return J
Initialize swarm arbitrary
      c1,c2,w,k,p
      Dimensions
      Number of particles
Call instance of PSO
Perform optimization
Create instances of Machine learning classifier
Get the selected features from the final positions
X_selected_features =  t[:,pos==1]
Perform classification and store performance in P
classifier.fit(X_selected_features, y)
Compute performance
subset_performance = classifier.predict(T[:,pos==1])
```

Whilst initializing the swarms arbitrary the custom functions that are used is as follows (Table1):

Table 1: PSO Parameters for Dataset1 and Dataset2.

Parameters	Dataset 1	Dataset 2
c1	0.5	1
c2	0.5	0.5
w	2.2	1
k	20	20
p	2	2

3) Genetic Algorithm

Furthermore, the Genetic Algorithm is a Heuristic Algorithm that is based on Darwin's theory of evolution. It is described as an Evolutionary Algorithm which finds the optimal solution in the process of natural selection and crossover. GA randomly generates individuals to produce an initial population. Each individual consists of a variable Gene which signifies a solution to the specific problem and is encoded by a chromosome.

The design of the GA contains three important operators:

- Selection
- Crossover
- Mutation.

"The selection operator is the method of selecting next-generation individuals from the current generation. The selection operator is designed to select individuals with the highest Fitness Values and remove bad solutions. The crossover is the method of gene recombination that recombines two parents' chromosomes to generate new individuals to be used in the next generation. The mutation operator is the process of modifying one or more gene values that are randomly selected within the current chromosome. The generational process based on genetic operators is repeated to gradually evolve candidate solutions

which converge on approximate solutions more and more. When the Genetic Algorithm process is terminated due to the given constraints, the optimum of the solutions is obtained to solve the problem"[11].

GA implementation can be seen in Figure 2.

Figure 2: Genetic Algorithm (GA) Implementation

ALGORITHM 2: GA

```
X_train ← training samples of textual data after pre-processing
X_test ← testing sample
y_train ← labels corresponding to training data.
y_test ← labels corresponding to testing data.
Def GA:
  P = Population Size
  F = Fitness Score
  S = Selection
  C = Crossover
  M = Mutations

Def generations (size,n_feat,n_parents,mutation_rate,n_gen,X_train, X_test, y_train, y_test):
   best_chromosome= []
   best_score= []
   NextGen_Population=initilization_population(size,n_feat)
   for i in range(n_gen):
      set Population_After_Fit to fitness_score(NextGen_Population)
      set Population_After_Selection to selection(Population_After_Fit,n_parents)
      set Population_After_Cross to crossover(Population_After_Selection)
      set NextGen_Population to mutation(Population_After_Cross,mutation_rate)
      best_chromosome.append(Population_After_Fit [0])
      best_score.append(scores[0])
   return best_chromosome,best_score

  set LR to LogisticRegression()
  LR.fit(X_train,y_train)
  set predictions to LR.predict(X_test)
  chromo,score=generations(size,n_feat ,n_parents
  ,mutation_rate,
                  n_gen
  ,X_train=X_train,X_test=X_test,y_train=y_train,y_test=y_test)
  LR.fit(X_train[:,chromo[-1]],y_train)

  set predictions to LR.predict(X_test[:,chromo[-1]])
End
```

G. *Transformers*

As previously stated, NLP was also implemented. Transformers' are credited to their abilities of understanding language to attain innovative results in NLP tasks. Due to Transformer models having transformed Natural Language Processing. BERT has swept away previously set benchmarks in nearly every common NLP task. This success, and the ensuing popularity, has seen the development of a whole host of new models based on the BERT architecture and training techniques. This research will not be attempting to choose the best model but rather will aim to provide a comparison between several of the most common ones, particularly in terms of accuracy. This paper analyses the efficacy of RoBERTa, DistilBERT, and XLNet pre-trained transformer models.

1) *RoBERTa*

The Robustly Optimized BERT approach or more commonly known as RoBERTa was established by the Facebook research team in 2019. It was introduced as an alternative optimized version of BERT, since the hyperparameters of BERT were tuned and it is retrained on a dataset ten times large. The masking strategy of BERT is changed from static to dynamic [12].

RoBERTa removes the Next Sentence Prediction (NSP) objective and adds dynamic masking of words during the training epochs. Such changes and features identify an enhanced performance compared to BERT in many NLP tasks, including text classification. Thus, RoBERTa uses more data to train, increases the batch size, removes next predict loss, and replaces static masking with dynamic masking in the pre-training stage to further improve the performance [12].

2) *DistilBERT*

DistilBERT was formed by applying knowledge distillation to BERT. It is based on a method that decreases the size of a BERT model by 40%, while maintaining 97% of its language understanding abilities and being 60% faster. To create a smaller version of BERT, DistilBERT creators removed token-type embeddings and the pooler from the architecture and reduced the number of layers by a factor of 2.

It is described that "The compact (student) model is trained to reproduce the full output distribution of the larger (teacher) model or ensemble of models. Rather than training with a cross-entropy over the hard-targets (one-hot encoding of the classes), the student obtains the knowledge based on a distillation loss over the soft-target probabilities of the teacher"[13].

The way DistilBert and RoBERTa were used can be seen in Figure 3.

Figure 3: DistilBERT/RoBERTa Implementation

ALGORITHM 3: DISTILBERT/ROBERTA

```
Implement tokenizer
Identify sequential length
Identify max length
Def tokenize
      for I, text in enumerate(df["tidy_tweet"]):
                tokens = tokenizer.encode_plus()
                input_ids.append()
                attention_masks.append()

      return  ((input_ids),  (attention_masks))
print(train_input_ids)
Def ecode labels:
     sentiment_values = set(df["label"].values)
       labels = []
       for index, row in df.iterrows():

          label = np.zeros((len(sentiment_values)))
          label[row["label"]] = 1
          labels.append(label)
     labels = np.asarray(labels)
     return labels
Implement RoBERTA/DistilBERT
Identify the neural network
embeddings = Roberta(input_ids, attention_mask=mask)
```

- LSTM(embeddings)
- BatchNormalization()
- Dense
- Activation("relu")
- Dense
- Dropout
- Dense(2, activation='softmax', name='outputs')

```
Identify optimizer, loss and acc
Fit model
Predict testing data
```

3) *XLNET*

Developed by Google Brain and Carnegie Mellon University, XLNet was created to enhance two aspects of BERT: Independence from masked tokens and "Pretrain-finetune discrepancy in training vs inference" [12]

These changes enabled to improve the architectural design for pre-training and produced results that outperformed BERT in various tasks. XLNet is described as a "auto regressive model" meaning that it does not require to mask certain tokens. To overcome the pretrain-finetune discrepancy Yang et al., proposed the Permutation Language Model (PLM) objective so it can capture bidirectional context by maximizing the expected log-likelihood of a sequence. Therefore, XLNet enhances the contextual information of each position by influencing tokens from all the other positions found on the left and right side of the token [14].

XLNET was implemented by:

1. Using XLNET tokenizer to convert the tokens to their index number in the XLNET vocabulary.
2. Pad input tokens and create attention masks.
3. Select a batch size for training. For fine-tuning with XLNet
4. Create an iterator of the data with torch DataLoader. This helps save on memory during training.
5. Load XLNEtForSequenceClassification, the pretrained XLNet model with a single linear classification layer on top.
6. Implement optimizer: AdamW
7. Prediction and Classification results

H. Performance Evaluation

To evaluate the performance of the different classifiers implemented as well as the transformer models used, four evaluation metrices are determined, more specifically accuracy, precision, recall and f1. By implementing the performance

techniques, the results will provide a deeper understanding on the behaviour of the different classifiers by identifying their strengths and weaknesses.

Moreover, the metrics will be defined in the terms of true and false positives [12]:

Accuracy : total number of correct predictions

$$Accuracy = \frac{\text{True Positive (TP)} + \text{True Negative(TN)}}{\text{Total Number of Observations}}$$

Precision: The ability of a classifier to not label a positive when in fact is negative.

$$Precision = \frac{true\ positive}{true\ positive + fasle\ positive}$$

Recall: The proportion of actual positive being identified correctly is given by recall

$$Recall = \frac{true\ postive}{true\ positive + false\ negative}$$

F1 score: Weighted mean of the Precision and Recall.

$$F1 = 2\ x\ \frac{precision * recall}{precision + recall}$$

III. EVALUATION OF METHODS

1) Setup

The experiments conducted for this work were completed using Python 3.7.4 running on Anaconda as it provides the advantage of using Jupyter Notebook for implementing the programs. Additionally, to conduct the NLP techniques google Colab is used since it speeds up the training and testing process for the multiple datasets. Colab allows anybody to write and execute arbitrary python code through google browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs

The project was conducted on a standard laptop, with 16 GB RAM and Interl® Core™ i7-10510U CPU @ 1.80GHz. Modules such as Scikit-Learn were implemented. Scikit-Learn is environment that is incorporated with Python programming language. The library offers a wide range of supervised algorithms that will be suitable for research. The library offers high-level implementation to train with the 'Fit' methods and 'predict' from an estimator (Classifier).

Furthermore, Keras and TensorFlow is used. Keras is an API that supports Neural Networks. The API supports other deep learning algorithms for easy and fast approach. It offers CPU and GPU running capabilities in order to simultaneously process the models. TensorFlow is an end-to-end ML platform that is developed by Google. The architecture lets a user run the program on multiple CPUs and it also has access to GPUs. TensorFlow can be incorporated with Keras to perform deep learning experiments.

2) Results and Discussion

The first stage of the research was to implement the data pre-processing, which enabled the raw data found in the datasets to be cleaned, becoming more understandable by the Machine Learning model. Both datasets were then split into training and testing samples after using CountVectorizer and TfidfTransformer using a 75%/25% split criterion. By implementing the SMOTE and RUS technique enabled the datasets go from being highly imbalanced to having equal classes. The reason to using both techniques was to not over generate synthetic instances, by removing samples from the over-classified label. Prior to applying the balancing method, the datasets had the following samples for each class:

- Dataset 1: 4,162 instances that were non-bullying (denoted as 0) and 1430 that were bullying (denoted as 1)
- Datase4 2: 11984 instances that were non-cyber bullying (denoted as 0) and 800 that were bullying (denoted as 1)

The data shows the need for more balanced classes, as the models used could produce a high accuracy but extremely low F1 score, meaning that the classifiers are producing countless false-positive results.

The third stage looked at the implementation of PCA. The role of PCA is to take a single text variable as an input and return numeric variables that summarize the text data, as well as tables of loadings to facilitate interpretation. This can be used either to provide a summary of text data, or, as an input to further analyses. PSO and GA were then both tested using Logistic Regression.

Moreover, RoBERTa, XLNet and DistilBERT were employed separately to identify how NLP performs in areas of cyberbullying. Table 2 and Table 3 illustrates the overall accuracy of the ML Framework and the NLP techniques. As it can be identified, the proposed model increased the accuracy of LR within both datasets by between 1-3%.

A comprehensive evaluation of these matrices with different machine learning classifier's performance is clarified in Table I and Table II. The paper used a different set of models and applied the Precision, Recall, F-Measure, and Accuracy individually for each. It can be concluded that ML approaches performance under the confusion matrices is gradually lower in regards to NLP, but this is expected due to the neural network being applied within the transformer models. As in real-time,

imbalanced class distribution exist in classification problems, regardless of using balancing methods; however, this can be tackled by F-Measure due to the utilization of crucial values of False Negative and False Positive.

Table 2: Dataset 1 Results

Classifier	**Dataset 1**			
	Accuracy	***Precision***	***Recall***	***F1-Score***
Machine Learning Framework				
LR	85.90%	86.43%	85.30%	86.11%
PCA-GA-LR	88.34%	88.47%	88.34%	87.58%
PCA-PSO-LR	86.98%	87.27%	86.98%	85.90%
NLP				
DistilBERT	86.78%	87.93%	86.77%	85.17%
XLNet	87.78%	87.50%	82.58%	83.71%
RoBERTa	86.43%	87.18%	86.43%	85.36%

Table 3: Dataset 2 Results

Classifier	**Dataset 2**			
	Accuracy	***Precision***	***Recall***	***F1-Score***
Machine Learning Framework				
LR	85.51%	92.08%	85.51%	88.14%
PCA-GA-LR	86.58%	90.9%	86.58%	88.47%
PCA-PSO-LR	85.64%	92.06%	85.64%	88.21%
NLP				
DistilBERT	94%	92.96%	94%	92.69%
XLNet	94.47%	94.43%	93.48%	93.88%
RoBERTa	93.89%	94.26%	93.89%	91.34%

As it can be observed within dataset machine learning classifiers works best while using Genetic Algorithm. Regardless, both techniques enabled to increase the findings of Logistic Regression. Furthermore, XLNet provided the highest outcomes in comparison to the other NLP methods (RoBERTa and DistilBERT) and the multi-stage Optimized Machine Learning framework between both datasets. This is due to XLNet being a large bidirectional transformer that uses improved training methodology, larger data, and more computational power to achieve better results.

IV. CONCLUSION

The project successfully implemented machine learning models in combination with bio-inspired algorithms and NLP techniques. The cyberbullying corpus used were in the form of text data, with approximately 20,000 “non-hate” and “hate” instances being evaluated with the proposed methodologies. Initially, Jupyter was used to apply various modules in the pre-processing stage to produce a cleaner set of data that would help during classification. Furthermore, Colab was utilized to conduct a combination of SMOTE and RUS as well as feature selection using PCA. The data was then passed to the classifier Logistic Regression and optimized using PSO and GA. This was done with the help of the Scikit-learns library as well as Pyswarms. The results identified that the multi-stage framework outperformed a simple machine learning classification. When comparing the two methods, we saw GA outperforming PSO. The second stage looked at NLP techniques and how they differ from machine learning methods, and as expected XLNet, RoBERTa and DistilBERT had more of an impact and outperformed the optimized Machine learning framework. However, there were situations where the proposed multi-stage optimized framework provided a higher F1 score, especially with the first dataset.

V. REFERENCES

[1] J.M. Xu, X. Zhu, and A. “Bellmore. Fast Learning for Sentiment Analysis on Bullying”. In WISDOM, 2012.

[2] S. Hinduja and J. W. Patchin, “Cyberbullying: An exploratory analysis of factors related to offending and victimization,” Deviant Behav., vol. 29, no. 2, pp. 129–156, 2008.

[3] C, Petrov “50 Alarming Cyberbullying Statistics for 2021”, https://techjury.net/blog/cyberbullying-statistics/, 2021.

[4] T. Davidson, D. Warmsley, M. Michael and I. Weber, Ingmar “Automated Hate Speech Detection and the Problem of Offensive Language” Proceedings of the 11th International AAAI Conference on Web and Social Media, pp.512-515

[5] K. Reynolds, A. Kontostathis and L. Edwards, "Using Machine Learning to Detect Cyberbullying", 2011 10th International Conference on Machine

[6] L.G., Kabari, B.,Nwamae," Principal Component Analysis (PCA) - An Effective Tool in Machine Learning" in International Journals of Advanced Research in Computer Science and Software Engineering, vol. 9, pp 56-59.

[7] M. Mafarja, R. Jarrar, S. Ahmad, and A. A. Abusnaina, "Feature selection using binary particle swarm optimization with time varying inertia weight strategies," in Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, 2018

[8] Sanyal , G., Ghosh,M.,: An ensemble approach to stabilize the features for multi-domain sentiment analysis using supervised machine learning. Journal of Big Data, pp. 1-25, 2018

[9] M. Mafarja, R. Jarrar, S. Ahmad, and A. A. Abusnaina, "Feature selection using binary particle swarm optimization with time varying inertia weight strategies," in Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, 2018

[10] Readthedocs.io. [Online]. Available: https://pyswarms.readthedocs.io/en/develop-ment/examples/featuresubsetselection. [Accessed: 11-Jun-2021]

[11] S.-S. Hong, W. Lee, and M.-M. Han, ''The feature selection method based on genetic algorithm for efficient of text clustering and text classification,'' Int. J. Adv. Soft Comput. Appl., vol. 7, pp. 2074–8523, Mar. 2015

[12] M. Aßenmacher and C. Heumann, "On the comparability of Pre-trained Language Models," arXiv [cs.CL], 2020.

[13] K. Mishev, A. Gjorgjevikj, I. Vodenska, L. T. Chitkushev, and D. Trajanov, "Evaluation of sentiment analysis in finance: From lexicons to transformers," IEEE Access, vol. 8, pp. 131662–131682, 2020.

[14] M. Khalid, I. Ashraf, A. Mehmood, S. Ullah, M. Ahmad, and G. S. Choi, "GBSVM: Sentiment classification from unstructured reviews using ensemble classifier," Appl. Sci. (Basel), vol. 10, no. 8, pp. 2788, 2020.