DOI: http://dx.doi.org/10.26483/ijarcs.v16i6.7360

Volume 16, No. 6, November-December 2025

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

CASE STUDY AND REPORT

Available Online at www.ijarcs.info

EMOTIONAL DETECTION ON SOCIAL MEDIA USING AI: A CASE STUDY OF X (FORMERLY TWITTER)

Faisal Abdullah Althobaiti
Department of Information Technology
King Abdulaziz University,
Jeddah, Saudi Arabia
ORCID - https://orcid.org/0009-0003-4517-1524

Abstract: The advent of the digital era increased the use of social media platforms such as X, which has also become a platform where humans express their emotions and generate a large volume of complex emotional data. This research explores artificial intelligence (AI) applications, especially natural language processing (NLP), for emotion detection in text on social media platforms. Advanced classifiers like SVM, RNN, LSTM, CNN, and GRU are used to study the performance and identify emotion spectrums beyond binary sentiment analysis. Comparative analysis of before and after optimization outcomes shows the significance of hyperparameter tuning and cross-validation in enhancing the model metrics like accuracy and F1-score. GRU is the top after post-optimization, with 93.10% accuracy and outstanding generalization. On the other hand, CNN is also strong, with 92.44%. Based on this, the significance of optimization techniques is figured out. Thus, GRU and CNN are optimal options for emotion detection on X because the platform gives strong support to mental health, marketing and public sentiment analysis.

Keywords: Emotional Detection, Social Media, Natural Language Processing (NLP), Sentiment Analysis, Deep Learning, RNN, CNN, GRU, Hyperparameter Tuning, Cross-Validation.

I. INTRODUCTION

The digital age brought about the proliferation of social media platforms like Facebook, Instagram, and X (formerly known as Twitter). These, among many others, are now part of how people emote, opine and think. The platforms generate a large volume of text data daily, including capturing many emotional expressions with a unique opportunity for massive analysis of human behavior [1]. This array of data generated empowers researchers with the know-how of real-time emotional study and the application of insights across boards, such as its application in mental health, marketing, and social research.

The development of artificial intelligence (AI) has changed how data is analyzed and interpreted by researchers, especially when it comes to how humans interact [2]. For instance, [2] further states that AI has computational power and top-notch algorithms needed to process voluminous text data efficiently. A major advancement in this is the deployment of Natural Language Processing (NLP). This is one of the most specialized parts of AI, whose focus is to enable machines to understand, interpret and generate data as humans do [3]. As shown in the findings of [4], NLP bridges the gap between human communication and machine learning systems, changing unstructured and spurious text data to structured ones that machines can easily process and analyze.

Although past studies such as [5], [6], [7], explored emotion detection on social media platforms, their focus was basically on binary sentiment analysis, which groups content as positive or negative sentiments. Human emotions are complex, consisting of a broad spectrum such as anger, fear, happiness, sadness, and surprise [8].

X, as a social media platform, is crucial to individuals' emotional expressions, opinion sharing and communication [9]. The large expanse of data generated by the platform is an opportunity for researchers to explore variability in human emotions at a larger scale. No doubt, different studies have been done a lot in the past on the subject matter. However, the focus was on binary sentiment analysis [10], [11]. This kind of analysis oversimplifies the inherent complexities in human emotions, especially because emotions need a more robust detection framework for analysis and drawing of conclusions [12]. Thus, the existing studies have not addressed the inherent challenges in informal, context-dependent languages usually used in social media.

Also, the tools and methodologies deployed by researchers for emotion detection in social media are not rigorous in terms of the effectiveness of their classifiers [13] and they have not succeeded in harnessing the potential of AI-driven models like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM). Therefore, a sophisticated framework that integrates a robust framework for a thorough evaluation of classifiers is imperative to address these gaps. Hence, this study.

The research questions for this study are as follows:

- Which classifier (SVM, RNN, LSTM, CNN, or GRU) provides the most accurate and reliable results for emotion detection in social media text combined with a robust data preprocessing pipeline?
- In what ways can hyperparameter tuning strengthen the AI-based emotion detection system's performance and scalability if applied in different industries?

Since binary sentiment analysis is limited, this study, therefore, advances academic research by introducing a robust framework to detect a wider range of emotions in social media text. It, therefore, contributes to the growth of the body of knowledge in AI, NLP and social media analytics. The study is also significant in that there are real-time implications in different fields; for instance, this study helps early detection of emotional distress to provide prompt intervention. Also, the emotion-based analysis of consumer sentiment can enhance targeted marketing strategies.

The contributions of this study are provided below.

- Classifier evaluation: The current study compares CSS, RNN, LSTM, CNN, and GRU classifiers to discover which is most effective for detecting emotion in social media text.
- Performance optimization: Comprehensive solutions can be provided for processing spurious and context-dependent data, as hyperparameter tuning can enhance the classifier's accuracy and scalability.
- Real-world applications: The researcher shows practical case studies of how the subject matter is applied to conduct mental health, marketing, and public sentiment analysis. This way, academic research can meet the needs of industry.

II. Literature Review

A. Approaches to emotion detection

Detecting emotions from text encompasses the identified and classified emotional expression found in the written text [14]. Over time, researchers were able to use different approaches, such as traditional machine learning, deep learning models, and specialized classifiers, and each of these had its inherent benefits and weaknesses [15].

Machine Learning Approaches

This approach is one of the early efforts of emotion detection, which depended on machine learning techniques like Support Vector Machines (SVM), Naive Bayes, and Random Forests [14], [16]. These models needed to be handcrafted with features obtained from sentiment lexicons, n-grams, and Part-of-Speech (POS) tagging [17]. Although they were effective for structured datasets, their limitations are due to overdependence on set rules and feature engineering; as such, their adaptability to informal and diverse social media text is limited [18].

For instance, SVM is robust for binary classification tasks and small datasets; however, it does not do well with voluminous, high-dimensional datasets, which social media generates [19]. Naïve Bayes, on the other hand, despite its computational efficiency, still does not perform efficiently when capturing context and sequential dependencies in text [20]. These limitations are felt more in spurious and unstructured text due to the presence of slang, emojis and abbreviations, limiting its overall performance.

Deep Learning Approaches

This model has changed the way emotions are detected, as it mitigated the need for complex feature engineering. It is the type that performs well at identifying inherent complex patterns in text [21]. Architectures like Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNN) have performed excellently [22].

- RNN and LSTM: These models perform well with sequential data such as text. They retain contextual information across time steps [23]. LSTM networks tackle the issue of vanishing gradients in standard RNNs; because of this, they are highly effective in identifying long-term dependencies [24]. For example, LSTM understands emotional context in sentences where different words are implied, such as "I'm not happy about the outcome."
- CNN: At first, this model only processed images traditionally. However, due to advancements, it now classifies text through pattern identifications in n-grams and hierarchical feature extraction [25]. CNNs are highly effective when it comes to detecting text with local dependencies [26].
- Transformer-based models: BERT and GPT-3 are two of the advanced models in this field. They use attention mechanisms to identify global dependencies in text [27]. These models perform effectively when identifying language complexities like sarcasm and ambiguity, which are prevalent on social media.

Deep learning models are unique because they are effective with transfer learning. They can also fine-tune the pre-trained models on specific datasets, enhancing task performance significantly even if the labeled data is limited.

Specialized Classifiers

Contextual Semantic Similarity (CSS) and Gated Recurrent Unit (GRU) are significant examples of emerging specialized classifiers, and they have brought another dimension and level of sophistication into emotion detection [28]. Their focus is on semantic relationships and vector-based representations of text, which offer comprehensive performance in unstructured and diverse datasets [29].

- CSS: CSS identifies and captures subtle emotional expressions inherent in text by emphasizing contextual understanding. For example, it can distinguish between the expression of sadness and happiness even when expressed in the same words [30], for instance, "I'm crying" (sadness) and "I'm crying tears of joy" (happiness) because of its ability to analyze a broader context semantically.
- GRU: GRU uses gating mechanisms for efficient capturing of sequential dependencies in text data, which then makes it appropriate for the identification of emotional group. The use of bidirectional layers makes GRU process input sequences from different directions, which enhances its identification ability for emotional patterns. Its ability to handle different sentence structures and style makes it more effective, which is an added advantage for social media data analysis, where tone and expression significantly varies [31].

B. Applications of Emotion Detection

Emotion detection has become crucial to research due to its application in different domains of study. The findings of emotional expression analysis in text data can present insightful information on how individuals and groups behave. Thanks to emotion detection systems, areas such as healthcare and business intelligence have become more impactful. For instance,

Mental health monitoring: The emotion detection system can unravel underlying mental health issues in people based on their social media posts and messages [32]. Thus, negative trends in emotional expressions can be promptly identified. Also, government and health organizations use the system for emotional health assessment during pandemics or major disasters [33]. Emotion-aware chatbots can be integrated into mental health platforms to provide empathetic and personalized support [34].

Marketing and consumer insights: Business organizations can use emotion detection systems to analyze customer sentiment [35]. The findings from such analysis can be used to develop an optimized strategy. Information from social media reviews can be leveraged to measure consumer satisfaction and address their concerns [36]. Also, tailored campaigns can be used to evoke excitement, a positive emotion that enhances buying intention.

Public sentiment monitoring: Government and other policymakers can better understand people's opinions on various issues affecting them using an emotion detection system [37]. For instance, sentiment during special events like elections and strike actions can be easily analyzed to get real-time information. Also, policymakers with the system can easily assess the emotional response to policies and attendant issues. The findings of such analysis can be used to refine people-friendly strategies.

Education and E-learning: In this field, the emotion detection system adds and enhances personalized learning experiences, improving student performance. For instance, the system can identify if students are bored or frustrated with content delivery [38]. This may mean the content needs to be augmented so they can understand better. Also, e-learning platforms leverage emotion-aware AI to streamline content and pace using students' emotional states. This promotes enhanced comprehension and facilitates retention [39].

The versatility of emotion detection tools has far-reaching applications in different domains. Incorporating the information generated into human emotions helps with making an informed decision, enhances user experiences, and contributes to the well-being of society [40]. Continuous development in the field and advancements in AI and NLP will still refine emotion detection systems, expanding their applicability and impact [41].

C. Performance Optimization

The optimization of emotion detection system performance is critical in ensuring the system's accuracy, efficiency, and scalability, especially when it comes to spurious and unstructured social media data [42]. Performance optimization is all about model architecture enhancement, hyperparameter fine-tuning, and leveraging advanced techniques to improve evaluation metrics like accuracy, precision, recall, and F1-score [43].

1. Importance of Optimization in Emotion Detection

The emotion detection model is operational on voluminous and complex datasets whose features are informal language, abbreviations, emojis, and contextual ambiguity [44]. Models may be unable to generalize effectively without optimization, which could result in poor performance classification [44]. Some of the features of performance optimization include:

• High accuracy: The system correctly classifies anger, happiness, sadness, and surprise, which are mixed emotions, in different text inputs [45].

• Efficiency: It reduces computational overhead in real-time social media stream processing [45].

• Scalability: The model can handle large datasets without affecting performance [46].

2. Hyperparameter Tuning

This is crucial to machine and deep learning model optimization. It encompasses parameter adjustments that control the learning process and enhance the model's performance [47]. Common hyperparameters in the system include:

- Learning rate: This determines the step size in the algorithmic gradient descent. An effectively optimized learning rate would ensure rapid convergence without over flogging the solutions provided [48].
- Batch size: This controls the volume of training examples deployed into an iteration. Even though smaller batch sizes may sometimes be spurious, they provide faster feedback [48]. Large batches, on the other hand, are stable but take more time to compute.
- Dropout Rate: This is about dropout regulation in neural networks for prevention of the problem of overfitting by randomly deactivating neurons during training [48].
- The number of layers and neurons determines the model's complexity. For example, adding layers in RNNs and LSTMs could enhance the model's ability to identify temporal dependencies, yet the risk of overfitting could rise [22].

Techniques for Hyperparameter Tuning:

Grid search: This is a systematic search using predefined hyperparameter values. It is expensive to compute; however, if successfully done, it makes the system comprehensive [47].

Random search: The system chooses hyperparameter combinations at random, which makes it better than grid search [47].

Bayesian Optimization: The system uses probabilistic models to predict hyperparameter combination performance, reducing the number of iterations needed [48].

Automated tuning: Optuna and Hyperopt tools can automate the tuning process, which uses advanced algorithms to identify optimal configurations [48].

Research Gap

Many studies explored different approaches to emotion detection; however, significant gaps remain in addressing social media text's inherent complexities and unique challenges. Machine learning frameworks like SVM and Naïve Bayes still struggle with the informal nature of social media data, which includes slang, emojis and abbreviations. LSTM and CNN are advanced fields of deep-learning models that can reduce the need for extensive feature engineering and the identification of long-term dependencies. Yet, there are limitations in understanding context-dependent emotional expressions, like sarcasm or subtle changes in sentiments.

Furthermore, models such as BERT and GPT-3 enhance the capabilities of identification of global dependencies and handling language complexities; their applications to different unstructured datasets have not been explored fully, especially with multilingual contexts. Developing specialized classifiers such as CSS and GRU is promising as they are effective across large-scale datasets. However, they still need further validation, which current literature often overlooks, particularly how hyperparameter tuning impacts the real-world application of emotion detection systems, like the scalability and computational efficiency of processing real-time social media streams.

III. METHOD

This study underscores the transformative impact of AI-driven emotion detection on X. Prior to optimization, SVM was the best-performing model because of its accuracy, F1 score, and specificity. However, post-optimization using hyperparameter tuning and cross-validation, CNN emerged as the best-performing model, achieving the highest accuracy and showing comprehensive generalization capabilities. These findings highlight the importance of optimization strategies for model performance enhancement, especially when it comes to handling imbalanced datasets and complex emotional classifications.

This study focuses on 7 major steps to complete emotional detection.

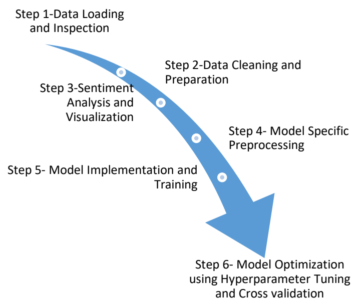
Step 1-Data Loading and Inspection
Step 2-Data Cleaning and Preparation
Step 3-Sentiment Analysis and Visualization
Step 4- Model Specific Preprocessing
Step 5- Model Implementation and Training
Step 6- Model Optimization using Hyperparameter Tuning and Cross validation

Fig 1. Methodology

The complete information about the steps completed is provided below.

Step 1: Data Loading and Inspection

pd.read_csv() function is used to load the dataset. This allows for proper structural and content exploration. The first ten rows are the data snapshot, so row counts indicate successful loading. Doing this strengthens familiarity with how the dataset is composed prior to processing. The dataset was downloaded from Kaggle [49].

Step 2: Data Cleaning and Preparation

Tweet_id is an example of unnecessary columns that are dropped to focus on suitable attributes such as content and sentiment. For dataset integrity, missing values are removed. Also, redundancy that can alter the results is identified and eliminated from the duplicate rows. The summary of the data cleaning is provided below.

```
First 10 rows of the dataset:
   Unnamed: 0                                               text  label
0           0      i just feel really helpless and heavy hearted      4
1           1  ive enjoyed being able to slouch about relax a...      0
2           2  i gave up my internship with the dmrg and am f...      4
3           3                         i dont know i feel so lost      0
4           4  i am a kindergarten teacher and i am thoroughl...      4
5           5       i was beginning to feel quite disheartened      0
6           6  i would think that whomever would be lucky eno...      2
7           7  i fear that they won t ever feel that deliciou...      1
8           8  im forever taking some time out to have a lie ...      5
9           9  i can still lose the weight without feeling de...      0

Data cleaning summary:
Number of rows after cleaning: 416123
Columns in the dataset: ['text', 'label']
```

Step 3: Sentiment Analysis and Visualization

At the post-dataset cleaning, analysis is conducted to understand the sentiment's frequency and distribution. Each sentiment occurrence is counted while its percentage values are calculated. With a bar chart, the result is visualized for sentiment prevalence highlight, which helps with the interpretation.

The dataset distribution can be seen in the figure below.

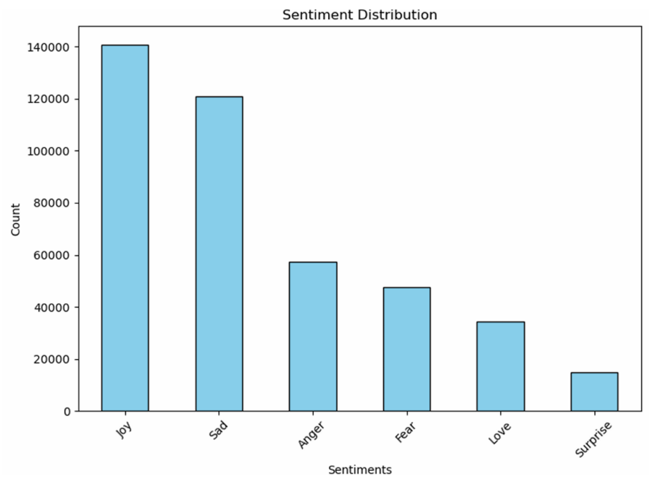

Fig 2. Dataset Distribution

There is a significant level of class imbalance relative to the sentiments in the data distribution. With more than 140000 samples, Joy had the highest representation, followed by Sad with 120000 samples. Anger and fear had 58000 and 50000 samples respectively, so they were moderately represented. Love had 35000 samples, while Surprise had 15000 samples. This imbalance shows that the model requires class weighting, oversampling, or undersampling as techniques needed for fair learning in all categories of sentiments.

Step 4: Model-Specific Preprocessing

The transformation of text data into a suitable numerical format for each machine-learning model is needed. The TfidfVectorizer changes SVM text into TF-IDF features, which identify the word's relevancy. RNN, CNN, LSTM and GRU text are tokenized into sequences for deep learning models to optimize the data for model architecture.

Step 5, and step 6 information is provided in the next section.

IV. Model Explanation

A. Model 1 - Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm that classifies and executes regression tasks. Its effectiveness can also be seen in text-based sentiment analysis because it has the ability to work well with high-dimensional data using kernel functions, even if the dataset is limited. This model is effective with binary or multi-class emotional classification tasks, particularly if emotional data is a feature represented as vectors that are derived from images, text and voice. The pseudocode for the SVM model implemented before optimization can be seen below.

Algorithm for Model 1 – SVM model implementation before optimization

Input
Training data: X_train_tfidf, y_train_tfidf
Test data: X_test_tfidf, y_test_tfidf

Output
Evaluation metrics: Accuracy, Precision, Recall, F1-Score, Confusion matrix visualization

```
1. # Initialize SVM Model
2. $svm\_model \leftarrow LinearSVC(C = 1.0, max\_iter = 50000, dual = False)$
3. # Train the SVM Model
4. $svm\_model.fit(X\_train\_tfidf, y\_train\_tfidf)$
5. # Predict on Test Data
6. $y\_pred\_svm \leftarrow svm\_model.predict(X\_test\_tfidf)$
7. # Compute Confusion Matrix
8. $conf\_matrix \leftarrow confusion\_matrix(y\_test\_tfidf, y\_pred\_svm)$
```

```
# Compute Evaluation Metrics
accuracy ← accuracy_score(y_test_tfidf, y_pred_svm)
precision, recall, f1, _ ←
  precision_recall_fscore_support(y_test_tfidf, y_pred_svm, aver
  ' weighted' )
```

Optimization of SVM

```
# Select a Sample of the Dataset
sample_fraction ← 0.5
X_sample, _, y_sample, _ ←
  train_test_split(X, y, train_size =
  sample_fraction, stratify = y)
# Create a Pipeline for Text Preprocessing, Classification
pipeline ← Pipeline([
  ('tfidf', TfidfVectorizer(stop_words =
  'english', ngram_range = (1, 3))),
  ('svd', TruncatedSVD(random_state = 42)),
  ('sgd', SGDClassifier(random_state = 42))
])
# Define Hyperparameter Grid
param_grid ← {
  'tfidf__max_features': [10000, 11000],
  'tfidf__min_df': [5, 7],
  'tfidf__max_df': [0.8, 0.9],
  'svd__n_components': [250, 350],
  'sgd__alpha': [0.0001, 0.001],
  'sgd__penalty': ['l2', 'elasticnet'],
  'sgd__loss': ['hinge', 'log']
}
# Perform Cross − Validation with Grid Search
grid_search ← GridSearchCV(pipeline, param_grid, cv =
2, scoring = ' accuracy' , verbose = 1)
grid_search.fit(X_sample, y_sample)
# Train − Test Split for Evaluation
X_train, X_test, y_train, y_test ←
  train_test_split(X_sample, y_sample, test_size =
0.2, stratify = y_sample)
# Train and Evaluate the Best Model
best_model ← grid_search.best_estimator_
best_model.fit(X_train, y_train)
y_pred ← best_model.predict(X_test)
# Calculate Performance Metrics
svm_accuracy ← accuracy_score(y_test, y_pred)
svm_cm ← confusion_matrix(y_test, y_pred)
svm_precision, svm_recall, svm_fscore, _ ←
  precision_recall_fscore_support(y_test, y_pred, average =
  ' weighted' )
```

The optimization of this model was done with pipeline that combines TF-IDF vectorization, reduced dimensionality with Truncated SVD and SGDClassifier-induced classification. All three stages of hyperparameters, there were some fine-tuning which uses grid search for performance maximization. Some of the components of TF-IDF parameters are the number of features (max_features), minimum document frequency (min_df), and maximum document frequency (max_df), which ensure retention of relevant terms while at the same time excludes overly common or rare terms. SVD parameters like the number of components (n_components) were tuned for dimensionality reduction and information retention. Regularization strength (alpha), penalty types (l2, elasticnet), and loss functions (hinge, log) were explored for the SGDClassifier for classification optimization.

Hyperparameters combination was evaluated with cross-validation grid search to identify the best-performing configuration. The sub-dataset was used for efficiency, and to stratify sampling that is maintained by class balance. The evaluation of the best model was done using different test sets to achieve enhanced performance with high accuracy, precision, recall, and F1-score. The optimization of SVM model revealed thorough generalization while it effectively captures important patterns inherent in the data. This was validated with confusion matrix visualization and classification metrics.

B. Recurrent Neural Network (RNN)

This belongs to a class of neural networks, and it is designed to recognize data patterns that are arranged sequentially. Because of the "memory," RNNs can recall the inputs from previous connections. RNNs model temporal dependencies; therefore, they are suitable for emotional tone analysis. The pseudocode for the RNN model implemented before optimization can be seen below.

Algorithm for Model 2 – RNN model implementation before optimization

Input
Training data: X_train, y_train
Test data: X_test, y_test

Output
Evaluation metrics: Accuracy, Loss, Confusion Matrix Visualization

```
# Initialize RNN Model
model ← Sequential([
  Embedding(input_dim = max_words, output_dim =
128, input_length = max_len),
  LSTM(128, return_sequences = True),
  Dropout(0.2),
  LSTM(64),
  Dropout(0.2),
  Dense(32, activation = 'relu'),
  Dense(len(sentiments), activation = 'softmax')
])
# Compile the RNN Model
model.compile(optimizer = 'adam',
        loss = 'sparse_categorical_crossentropy',
        metrics = ['accuracy'])
# Train the RNN Model
epochs ← 3
batch_size ← 32
history ← model.fit(X_train, y_train,
            epochs = epochs,
            batch_size = batch_size,
            validation_data = (X_test, y_test))
# Evaluate the RNN Model
test_loss, test_accuracy ← model.evaluate(X_test, y_test)
# Generate Predictions
y_pred ← np.argmax(model.predict(X_test), axis = −1)
# Compute Confusion Matrix
cm ← confusion_matrix(y_test, y_pred)
```

Optimization of RNN

```
# Initialize Stratified k − Fold Cross − Validation
fold ← 1
cv_accuracies ← []
# Perform Stratified k − Fold Cross − Validation
For train_index, val_index in kfold.split(X_padded, y) do
  Print "Training fold " + fold
  fold ← fold + 1
  # Split the data into training and validation sets
  X_train, X_val ←
 X_padded[train_index], X_padded[val_index]
  y_train, y_val ← y.iloc[train_index], y.iloc[val_index]
  # Build the RNN Model
  model ← Sequential([
    Embedding(input_dim = max_words, output_dim =
embedding_dim, input_length = max_len),
        Bidirectional(LSTM(128, return_sequences = True)),
        Dropout(0.3),
        Bidirectional(LSTM(64)),
        Dropout(0.3),
        Dense(128, activation = 'relu'),
        Dropout(0.3),
        Dense(64, activation = 'relu'),
        Dense(len(y.unique()), activation = 'softmax')
    ])
    # Compile the Model
    optimizer ← Adam(learning_rate = learning_rate)
    model.compile(optimizer = optimizer,
            loss = 'sparse_categorical_crossentropy',
            metrics = ['accuracy'])
    # Early Stopping Callback
    early_stopping ← EarlyStopping(monitor =
  ' val_loss ' ,
                    patience = 3,
                    restore_best_weights = True)
    # Train the Model
    history ← model.fit(
        X_train, y_train,
        epochs = epochs,
        batch_size = batch_size,
        validation_data = (X_val, y_val),
        callbacks = [early_stopping],
        verbose = 1
    )
    # Evaluate the Model on Validation Set
    val_loss, val_accuracy ←
  model.evaluate(X_val, y_val, verbose = 0)
    Append val_accuracy to cv_accuracies
```

The RNN model was hyperparameter tuned and cross-validated to optimize its performance and generalization. LSTM units 128 and 64, 30% rate of dropout, batch size, learning and embedding dimensions are the chosen hyperparameters for iterative experimentation to create an accurate balance and computational efficiency. Not only that, the Adam optimizer's rate of learning was also fine-tuned for enhanced convergence without loss minima overshooting.

To divide K-fold, cross-validation was deployed for the evaluation of the robustness of the model across series of sub-datasets. This approach accurately validated each of the folds, to ensure the performance of the model did not rely totally on a certain training-validation divides. The accurate mean cross-validation showed the level of consistency and reliability of the model. Also, quick stopping while the training hold can further strengthen the performance by stopping the training when validation loss stopped to improve, prevent, overfit and save computational resources. Such integrative techniques could bring about a efficiently generalized RNN model.

C. Long Short-Term Memory (LSTM)

They use a gating mechanism to control the flow of information. They are now able to tackle the vanishing gradient problems in standard RNNs. LSTMs identify complex emotional patterns in text or speech with long sequences.

Algorithm for Model 3 – LSTM model implementation before optimization

Input
Training data: X_train, y_train
Test data: X_test, y_test

Output
Evaluation metrics: Accuracy, Loss, Confusion Matrix Visualization

```
  # Initialize LSTM Model
  lstm_model ← Sequential([
    Embedding(input_dim = max_words, output_dim =
  128, input_length = max_len),
    LSTM(128, return_sequences = True),
    Dropout(0.2),
    LSTM(64),
    Dropout(0.2),
    Dense(32, activation = 'relu'),
    Dense(len(sentiments), activation = 'softmax')
  ])
  # Compile the LSTM Model
  lstm_model.compile(optimizer = 'adam',
          loss = 'sparse_categorical_crossentropy',
          metrics = ['accuracy'])
  # Train the LSTM Model
  epochs ← 3
  batch_size ← 32
  lstm_history ← lstm_model.fit(X_train, y_train,
                epochs = epochs,
                batch_size = batch_size,
                validation_data = (X_test, y_test))
  # Evaluate the LSTM Model
  lstm_test_loss, lstm_test_accuracy ←
   lstm_model.evaluate(X_test, y_test)
  # Generate Predictions for LSTM
  y_pred_lstm ←
   np.argmax(lstm_model.predict(X_test), axis = −1)
  # Compute Confusion Matrix
  lstm_cm ← confusion_matrix(y_test, y_pred_lstm).
```

Optimization of LSTM

```
  # Initialize Stratified k − Fold Cross − Validation
  fold ← 1
  cv_accuracies ← []
  # Define Learning Rate Scheduler
  Function lr_scheduler(epoch, lr):
    If epoch > 5 then
      Return lr ∗ 0.5
    Else
      Return lr

  lr_callback ← LearningRateScheduler(lr_scheduler)
  # Perform Stratified k − Fold Cross − Validation
  For train_index, val_index in kfold.split(X_padded, y) do
    Print "Training fold " + fold
    fold ← fold + 1
    # Split the data into training and validation sets
    X_train, X_val ←
   X_padded[train_index], X_padded[val_index]
    y_train, y_val ← y.iloc[train_index], y.iloc[val_index]
    # Build the LSTM Model
    model ← Sequential([
      Embedding(input_dim = max_words, output_dim =
  embedding_dim, input_length = max_len),
      Bidirectional(LSTM(128, return_sequences = True)),
      Dropout(0.3),
      Bidirectional(LSTM(64)),
      Dropout(0.3),
      Dense(128, activation = 'relu'),
      Dropout(0.3),
      Dense(64, activation = 'relu'),
      Dense(len(y.unique()), activation = 'softmax')
    ])
    # Compile the Model
    optimizer ← Adam(learning_rate = learning_rate)
    model.compile(optimizer = optimizer,
          loss = 'sparse_categorical_crossentropy',
          metrics = ['accuracy'])
    # Early Stopping Callback
    early_stopping ← EarlyStopping(monitor = ' val_loss ' ,
                    patience = 3,
                    restore_best_weights = True)
    # Train the Model
    history ← model.fit(
      X_train, y_train,
      epochs = epochs,
      batch_size = batch_size,
      validation_data = (X_val, y_val),
      callbacks = [early_stopping, lr_callback],
      verbose = 1
    )
    # Evaluate the Model on Validation Set
    val_loss, val_accuracy ← model.evaluate(X_val, y_val, verbose = 0)
    Append val_accuracy to cv_accuracies
```

Bidirectional architecture was utilized in capturing past and future dependencies of sequential data in LSTM model. Two LSTM layers such as 128 and 64 units were implemented to create a balance between complexity of the model and efficient computation. The addition of 30% dropout layer was done after each LSTM layer and full connection to layers for overfitting prevention. The wholly connected layers were 128 and 64 units with ReLU activation, and this was followed by a softmax output layer for different classes classification.

In evaluation of the robustness of the model across different data divides, cross-validation was used to have accurate and consistent validation. A learning rate scheduler dynamically altered the learning rate, dividing it into two after epoch 5 for convergence enhancement. Quick or early stopping was applied for the training termination when validation loss no longer improves. This therefore prevented overfitting. High-performing LSTM model was the result of the optimization. This is concluded as such because of cross-validation accuracies, test performance measures which include the accuracy, precision, recall, and F1-score), and confusion matrix visualization, as well as confusion matrix visualization.

D. Convolutional Neural Network (CNN)

These are used to process images or any other grid-like data. They use convolutional layers for data features and pattern detection. However, they were applied in the past to text by treating it as a dimensional grid of words or characters. CNN has the capacity to extract facial expressions or local patterns from text.

Algorithm for Model 4 – CNN model implementation before optimization

Input
Training data: X_train, y_train
Test data: X_test, y_test

Output
Evaluation metrics: Accuracy, Loss, Confusion Matrix Visualization

```
# Initialize CNN Model
cnn_model ← Sequential([
  Embedding(input_dim = max_words, output_dim = 128, input_length = max_len),
  Conv1D(filters = 128, kernel_size = 5, activation = 'relu'),
  MaxPooling1D(pool_size = 2),
  Dropout(0.2),
  Flatten(),
  Dense(64, activation = 'relu'),
  Dropout(0.2),
  Dense(len(sentiments), activation = 'softmax')
])
# Compile the CNN Model
cnn_model.compile(optimizer = 'adam',
          loss = 'sparse_categorical_crossentropy',
          metrics = ['accuracy'])
# Train the CNN Model
epochs ← 3
batch_size ← 32
cnn_history ← cnn_model.fit(X_train, y_train,
                    epochs = epochs,
                    batch_size = batch_size,
                    validation_data = (X_test, y_test))
# Evaluate the CNN Model
cnn_test_loss, cnn_test_accuracy ← cnn_model.evaluate(X_test, y_test)
# Generate Predictions for CNN
y_pred_cnn ← np.argmax(cnn_model.predict(X_test), axis = −1)
# Compute Confusion Matrix
cnn_cm ← confusion_matrix(y_test, y_pred_cnn)
```

Optimization of CNN

```
# Initialize Stratified k-Fold Cross-Validation
fold ← 1
cv_accuracies ← []
# Perform Stratified k-Fold Cross-Validation
For train_index, val_index in kfold.split(X_padded, y) do
  Print "Training fold " + fold
  fold ← fold + 1
  # Split the data into training and validation sets for this fold
  X_train, X_val ← X_padded[train_index], X_padded[val_index]
  y_train, y_val ← y.iloc[train_index], y.iloc[val_index]
  # Build the CNN Model
  model ← Sequential([
    Embedding(input_dim=max_words, output_dim=embedding_dim, input_length=max_len),
    Conv1D(filters=128, kernel_size=5, activation='relu'),
    MaxPooling1D(pool_size=2),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    GlobalMaxPooling1D(),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(len(y.unique()), activation='softmax')
  ])
  # Compile the Model
  optimizer ← Adam(learning_rate=learning_rate)
  model.compile(optimizer=optimizer,
         loss='sparse_categorical_crossentropy',
         metrics=['accuracy'])
  # Early Stopping Callback
  early_stopping ← EarlyStopping(monitor='val_loss',
                 patience=3,
                 restore_best_weights=True)
  # Train the Model
  history ← model.fit(
    X_train, y_train,
    epochs=epochs,
    batch_size=batch_size,
    validation_data=(X_val, y_val),
    callbacks=[early_stopping],
    verbose=1
  )
  # Evaluate the Model on Validation Set
  val_loss, val_accuracy ← model.evaluate(X_val, y_val, verbose=0)
  Append val_accuracy to cv_accuracies
```

Structured approach was deployed to optimize CNN model for performance enhancement and to ensure overall generalization. Number of filters (128 and 64), sizes 5 and 3

Kernel, 30% of rate of dropout and the rate of learning for Adam Optimizer were iteratively fine-tuned. Some of the architectures are embedding layers, multiple convolutional layers to extract features, max pooling of layers to reduce dimensionality and wholly connected layers for classification.

For the evaluation of the model using different multiple data splits, cross-validation was utilized to ensure unbiased performance against certain dataset divisions. Quick stopping was done so as not to overfit, halt training when validation no longer improve. The integration of such technique strengthened CNN model capabilities to extract insightful features and effectively classify the task without compromising the cross-validation folds consistency.

E. Gated Recurrent Unit (GRU)

This uses RNN architecture for the optimization of sequential data. The incorporation of Bidirectional GRU layers enhances the capturing of past and future dependencies in the input sequences, which enhance the extraction of features. Dropout and L2 regularization are techniques that can be applied to reduce or eradicate overfitting and ensure generalization. The gating mechanism of GRU reduces complexities inherent in computations compared to LSTMs without compromising their performance. This then highlight their efficiency at analyzing emotional, trends in large datasets. The model is effective for temporal patterns identification and sequential dependencies, which ensure strong performance irrespective of the multi-class complexities sentiment classification tasks.

Algorithm for Model 5 – GRU model implementation before optimization

Input
Training data: X_train, y_train
Test data: X_test, y_test

Output
Evaluation metrics: Accuracy, Loss, Confusion Matrix Visualization

```
# Initialize GRU Model
gru_model ← Sequential([
  Embedding(input_dim = max_words, output_dim =
128, input_length = max_len),
  Bidirectional(GRU(128, return_sequences = True)),
  Dropout(0.2),
  Bidirectional(GRU(64)),
  Dropout(0.2),
  Dense(64, activation = 'relu'),
  Dropout(0.2),
  Dense(len(sentiments), activation = 'softmax')
])
# Compile the GRU Model
gru_model.compile(optimizer = 'adam',
          loss = 'sparse_categorical_crossentropy',
          metrics = ['accuracy'])
# Train the GRU Model
epochs ← 3 batch_size ← 32
gru_history ← gru_model.fit(X_train, y_train,
                  epochs = epochs,
                  batch_size = batch_size,
                  validation_data = (X_test, y_test))
# Evaluate the GRU Model
gru_test_loss, gru_test_accuracy ←
 gru_model.evaluate(X_test, y_test)
# Generate Predictions for GRU
y_pred_gru ←
 np.argmax(gru_model.predict(X_test), axis = −1)
# Compute Confusion Matrix
gru_cm ← confusion_matrix(y_test, y_pred_gru)
```

Optimization of GRU

```
# Initialize Cross-Validation
fold ← 1
cv_accuracies ← []
# Perform Cross-Validation for GRU-based Model
For train_index, val_index in kfold.split(X_padded, y) do
  Print "Training fold " + fold
  fold ← fold + 1
  # Prepare Training and Validation Data
  X_train, X_val ← X_padded[train_index], X_padded[val_index]
  y_train, y_val ← y.iloc[train_index], y.iloc[val_index]
  # Build the GRU-based Model
  model ← Sequential([
    Embedding(input_dim=max_words,
output_dim=embedding_dim, input_length=max_len),
    Bidirectional(GRU(128, return_sequences=True)),
    Dropout(0.3),
    Bidirectional(GRU(64)),
    Dropout(0.3),
    Dense(128, activation='relu', kernel_regularizer='l2'),
    Dropout(0.3),
    Dense(64, activation='relu', kernel_regularizer='l2'),
    Dense(len(y.unique()), activation='softmax')
  ])
  # Compile the Model
  optimizer ← Adam(learning_rate=learning_rate)
  model.compile(optimizer=optimizer,
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
  # Early Stopping Callback
  early_stopping ← EarlyStopping(monitor='val_loss',
                    patience=3,
                    restore_best_weights=True)
  # Train the Model
  history ← model.fit(
    X_train, y_train,
    epochs=epochs,
    batch_size=batch_size,
    validation_data=(X_val, y_val),
    callbacks=[early_stopping],
    verbose=1
  )
  # Evaluate the Model on Validation Data
  val_loss, val_accuracy ← model.evaluate(X_val, y_val,
verbose=0)
  Append val_accuracy to cv_accuracies
```

The optimization of the GRU model was achieved with bidirectional architecture, which captured sequential pattern from forward and backward contexts. 128 and 64 units were the two GRU layers that were used for balancing the model's complexity and efficient computation. 30% of dropout layers was integrated after each GRU and dense layer for overfitting reduction. The dense layers of 128 and 64 units used ReLU activation and L2 regularization for continuous prevention of overfitting and enhance generalization. The output layer utilized softmax activation for different classes classification.

For model robustness and consistency, cross-validation was conducted to validate the accuracies of the fold. Quickly stopping the training when validation loss plateaued, reduced overfitting and extra cost of computation. The final test set evaluation revealed that the model is strong for generalization, with high accuracy, precision, recall, and F1-score. The performance metrics and confusion matrix visualizations are the confirmation of the ability of the model in effective handling of classification task.

V. Results and Discussion

A. Results observation before optimization

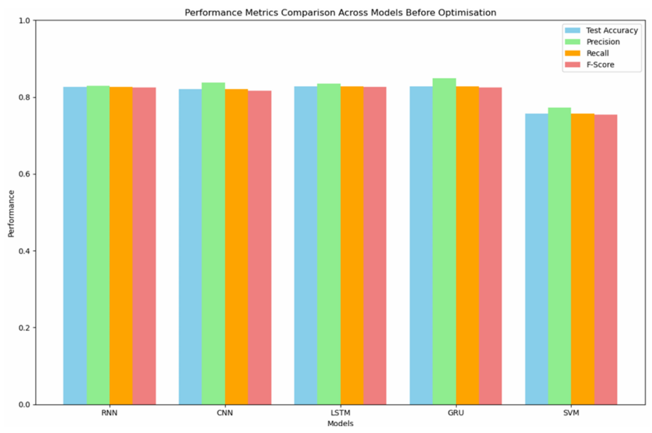

Fig 3. Results before optimization

0.8432 is the value for highest precision, while competitive test accuracy is 0.8279 and F-score is 0.8238. These values depict that GRU model performance is the best overall. This also implies a strong robustness of sequential data handling. The models that follow closely are RNN and LSTM with a bit of higher accuracy but lower precision than GRU. Also, the CNN Model performed reasonably, only lagging in sequential patterns handling. SVM model is the least of all with an accuracy value of 0.7571 and F-score, showing its limitation when it is compared to deep learning models.

B. Results observations after optimization

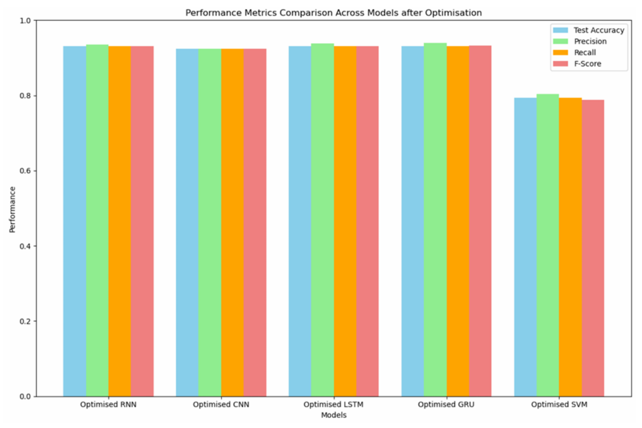

Fig 4. Results after optimization

This showed different performance levels; GRU emerged as the highest-performing model, with 93.10% test accuracy, 93.96% precision and 93.18% F-Score, which shows its ability for effective management of sequential data. This is followed by LSTM at 93.07% test accuracy and 93.8% precision. This makes it a robust choice for tasks that need long-term reliance. The CNN model may be less accurate at 92.44% test accuracy, yet it is a viable choice in cases where spatial feature extraction is important. Contrastingly, the SVM model had a test accuracy of 79.34% and an F-Score of 78.83%, which showed that it was the least effective when it came to handling complex, sequential datasets. In the overall, RNN models such as GRU and LSTM performed consistently and had generalization in different validation folds, as such they are the recommended options for sequential or time-series data activities.

C. Comparison of before and after optimization

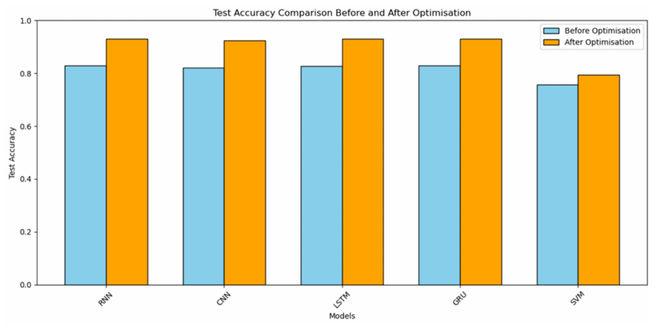

Fig 5. Accuracy of models before and after optimization

Test accuracy: Optimization resulted in significant improvement in test accuracy in all the models. RNN, for instance, improved the most, with an increase from 82.59% to 93.14%. This implies it has an enhanced capacity to generalize. This is followed by LSTM and GRU with 93.07% and 93.10% accuracy respectively. This shows that they are robust in terms of sequential data. CNN, on the other hand, increased from 82.06% to 92.44%, while the SVM also increased from 75.91% to 79.34%, a modest gain but still trailing behind deep learning models in the overall performance of all the models.

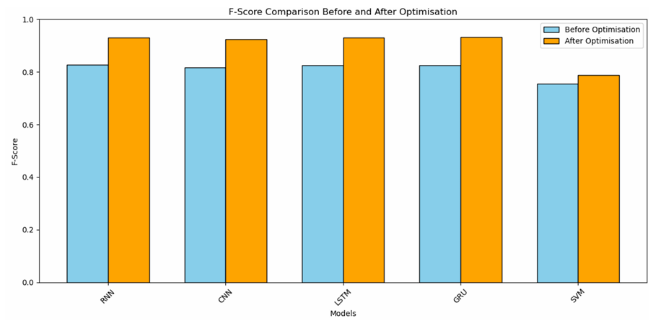

Fig 6. F1-score of the models before and after optimization

F-Score: This creates a balance between precision and recall and significantly improved all the model's post-optimization; for example, the GRU had the highest score, ranging from 82.49% to 93.18%. This is followed by LSTM ranging from 82.55% to 93.13%. Also, RNN and CNN recorded some levels of improvement, to the tune of 93.05% and 92.35%, respectively. Only SVM was modest, with an increase from 75.44% to 78.83%. However, it can be regarded as below-par performance if compared to deep learning models. This shows it has relative limitations in fostering a balance between precision and recall.

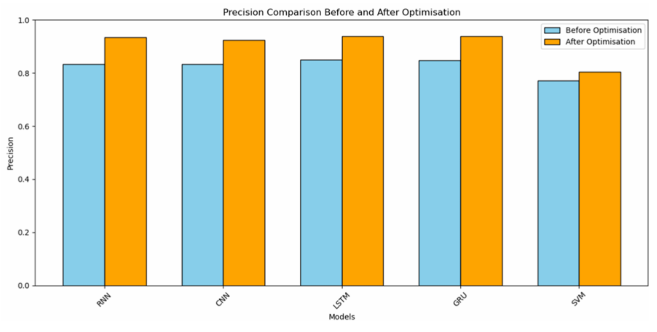

Fig 7. Precision of the models before and after optimization

Precision: Every model in this study has significant improvement in their precision after they have been optimized. For instance, both LSTM and GRU had the highest gain, with precisions ranging from 85.02% to 93.80% and from 84.70% to 93.96%, respectively. This is followed by RNN and CNN with an improvement ranging from 83.39% and 83.43% to 93.56% and 92.41%, respectively. SVM had the least improvement, with its performance declining from 77.20% to 80.37%. This is an indication that despite its enhanced performance optimization, it is not effective like the deep learning models.

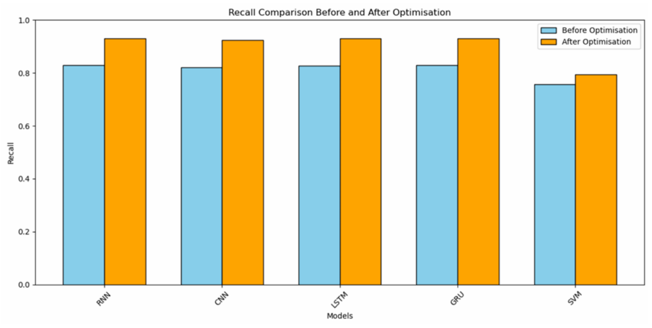

Fig 8. Recall of the model before and after optimization

Recall: Consistent optimization increased recall in all the models. This is a reflection of improved sensitivity. RNN, LSTM, and GRU all improved maximally, with recall being more than 93%. This is followed by CNN, with a significant rise from 82.06% to 92.44%. Conversely, the SVM had modest improvement ranging from 75.71% to 79.34%. All these are reflections of the superior capability of the deep learning model and its correct identification of positive cases compared to SVM.

D. *Final results*

Before the models were optimized, there were different performances. SVM had an accuracy of 0.7571; this was balanced with 0.7720 precision and 0.571 recall. RNN, LSTM and GRU were better off SVM, with 0.8280 accuracies. Only CNN recorded the lowest performance with a value of 0.8206. After optimization, GRU was the highest-performing model with an accuracy of 0.8206, 0.9936 precision and 0.9318 F1-Score. This is followed by RNN and LSTM with 0.9314 and 0.9307, respectively, coupled with balanced precision and recall. CNN was also accurate behind the GRU, RNN and LSTM with an accuracy of 0.9244 and 0.235 F1-Score. SVM recorded modest accuracy, that is, 0.7934. Based on this, GRU was the best overall, followed by RNN and LSTM.

E. *Discussion*

Emotional detection practical applications cut across different strata, including marketing, mental health and public sentiment analyses. Based on the information obtained from social media datasets, the application focuses on understanding the emotional trends of individuals and groups of individuals.

Emotion detection is significant for understanding consumer sentiment in marketing about campaigns, products and services. Positive sentiments like happiness and love trigger the satisfaction of the consumers, and they are the main part of the dataset. Even though they are 27% of the overall datasets, they are important feedback to organization to fine-tune their strategies. Negative sentiment, that is, worry and sadness, which are 21.8% and 12.93%, respectively, are equally important because they let brands address their concerns proactively, improve relationships with customers and let products match the needs of the consumers.

Emotion detection is significant with respect to monitoring the opinion of the public, where worry and neutral sentiments are the dominant emotions, and they are 43% of the overall datasets. They provide information with respect to the responses of society to current events or policies, which help policymakers understand and make informed decisions that effectively address public concerns. Other emotions, such as anger (0.28%) and enthusiasm (1.9%), also let the government and organization feel the pulse of the public and plan appropriate interventions.

Emotion detection in mental health analysis provides information as to the well-being of people. Sadness and relief, for example, take about 3.82% of the total datasets, and they are needed to monitor mental health. Similarly, boredom (0.45% and surprise (5.48%), which are not dominant emotions, help in solidifying the understanding and enabling the targeted development of mental health initiatives. The application of such shows the significance of in-depth emotional insights that are capable of ensuring the development of mental health programs.

CNN and GRU are advanced AI models with a significant impact on emotion detection performance, as they address the issue of class imbalance and improve important indices like precision, recall, and F-score. From the optimization result, CNN had 92.44% accuracy and 92.41% precision. GRU was exceptional also, with 93.10% accuracy and 93.95% precision. These indices show that the model is robust enough to offer dependable and various emotional insights across many domains. This is why they are significant tools for emotion detection application and optimization.

VI. Conclusion

This research has shown the transformative impact of AI-driven emotion detection across different areas such as mental health, marketing, and public policy. After the optimization, GRU was the overall best-performing model with 93.10% accuracy and an F1-Score of 93.18%. CNN improved maximally; as such, it is an important model in feature-driven activities. Optimization strategies were crucial to address the imbalances in the datasets and performance. RNN and LSTMs, which are sequential models, excelled in short-time pattern identification. These findings, therefore, have provided requisite tools to advance emotion detection and its practical applications.

VII. References

[1] E. Gutierrez, W. Karwowski, K. Fiok, M. R. Davahli, T. Liciaga, and T. Ahram, "Analysis of human behavior by mining textual data: Current research topics and analytical techniques," *Symmetry*, vol. 13, no. 7, p. 1276, 2021. [Online]. Available: https://doi.org/10.3390/sym13071276

[2] C. Collins, D. Dennehy, K. Conboy, and P. Mikalef, "Artificial intelligence in information systems research: A systematic literature review and research agenda," *International Journal of Information Management*, vol. 60, p. 102383, 2021. [Online]. Available: https://doi.org/10.1016/j.ijinfomgt.2021.102383

[3] F. Scarcello, "Artificial intelligence," *Encyclopedia of Bioinformatics and Computational Biology*, pp. 287–293, 2018. [Online]. Available: https://doi.org/10.1016/B978-0-12-809633-8.20326-9

[4] K. Kumar, M. Syed, S. Bhargava, L. Mohakud, *et al.*, "Natural language processing: Bridging the gap between human language and machine understanding," in *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*, Pune Lavasa Campus, India, Mar. 22–23, 2024, pp. 1–7. doi: 10.1109/TQCEBT57456.2024.123456.

[5] P. Nandwani and R. Verma, "A review on sentiment analysis and emotion detection from text," *Social Network Analysis and Mining*, vol. 11, no. 1, p. 81, 2021. [Online]. Available: https://doi.org/10.1007/s13278-021-00776-6

[6] J. R. Jim, M. A. R. Talukder, P. Malakar, M. M. Kabir, K. Nur, and M. Mridha, "Recent advancements and challenges of NLP-based sentiment analysis: A state-of-the-art review," *Natural Language Processing Journal*, vol. 6, p. 100059, 2024. [Online]. Available: https://doi.org/10.1016/j.nlp.2024.100059

[7] J. Y. Nip and B. Berthelier, "Social media sentiment analysis," *Encyclopedia*, vol. 4, no. 4, pp. 1590–1598, 2024. [Online]. Available: https://doi.org/10.3390/encyclopedia4040104

[8] M. B. Akçay and K. Oğuz, "Speech emotion recognition: Emotional models, databases, features, preprocessing methods, supporting modalities, and classifiers," *Speech Communication*, vol. 116, pp. 56–76, 2019. [Online]. Available: https://doi.org/10.1016/j.specom.2019.12.001

[9] D. G. Graciyal and D. Viswam, "Social media and emotional well-being: Pursuit of happiness or pleasure," *Asia Pacific Media Educator*, 2021. [Online]. Available: https://doi.org/10.1177/1326365X211003737

W. van Atteveldt, M. A. C. G. van der Velden, and M. Boukes, "The validity of sentiment analysis: Comparing manual annotation, crowd-coding, dictionary approaches, and machine learning algorithms," *Communication Methods and Measures*, vol. 15, no. 2, pp. 121–140, 2021. doi: 10.1080/19312458.2020.1869198.

[10] N. Braig, A. Benz, S. Vith, J. Breitenbach, *et al.*, "Machine learning techniques for sentiment analysis of COVID-19-related Twitter data," *IEEE Access*, 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10035946

[11] A. De Leon Langure and M. Zareei, "Improving text emotion detection through comprehensive dataset quality analysis," *IEEE Access*, 2024. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10744050 [Accessed: Nov. 28, 2024].

[12] P. Sarmah, R. Das, S. Dhamija, S. Bilgaiyan, and B. S. P. Mishra, "Facial identification expression-based attendance monitoring and emotion detection—A deep CNN approach," *Machine Learning for Biometrics*, pp. 155–176, 2021. [Online]. Available: https://doi.org/10.1016/B978-0-323-85209-8.00001-8

[13] K. Machová, M. Szabóova, J. Paralič, and J. Mičko, "Detection of emotion by text analysis using machine learning," *Frontiers in Psychology*, vol. 14, p. 1190326, 2023. [Online]. Available: https://doi.org/10.3389/fpsyg.2023.1190326

[14] M. M. Taye, "Understanding of machine learning with deep learning: Architectures, workflow, applications, and future directions," *Computers*, vol. 12, no. 5, p. 91, 2023. [Online]. Available: https://doi.org/10.3390/computers12050091

[15] F. Cavicchio, "Machine learning approaches to emotion detection," 2024. [Online]. Available: https://doi.org/10.1007/978-3-031-72047-5_5

[16] S. Chotirat and P. Meesad, "Part-of-speech tagging enhancement to natural language processing for Thai wh-question classification with deep learning," *Heliyon*, vol. 7, no. 10, p. e08216, 2021. [Online]. Available: https://doi.org/10.1016/j.heliyon.2021.e08216

[17] Y. Zhang and Z. Wang, "Feature engineering and model optimization based classification method for network intrusion detection," *Applied Sciences*, vol. 13, no. 16, p. 9363, 2022. [Online]. Available: https://doi.org/10.3390/app13169363

[18] R. Guido, S. Ferrisi, D. Lofaro, and D. Conforti, "An overview on the advancements of support vector machine models in healthcare applications: A review," *Information*, vol. 15, no. 4, p. 235, 2024. [Online]. Available: https://doi.org/10.3390/info15040235

[19] W. Y. Melhem, A. Abdi, and F. Meziane, "Deep learning classification of traffic-related tweets: An advanced framework using deep learning for contextual understanding and traffic-related short text classification," *Applied Sciences*, vol. 14, no. 23, p. 11009, 2023. [Online]. Available: https://doi.org/10.3390/app142311009

[20] N. Manakitsa, G. S. Maraslidis, L. Moysis, and G. F. Fragulis, "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision," *Technologies*, vol. 12, no. 2, p. 15, 2024. [Online]. Available: https://doi.org/10.3390/technologies12020015

[21] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: A comprehensive review of architectures, variants, and applications," *Information*, vol. 15, no. 9, p. 517, 2024. [Online]. Available: https://doi.org/10.3390/info15090517

[22] I. Malashin, V. Tynchenko, A. Gantimurov, V. Nelyub, and A. Borodulin, "Applications of long short-term memory (LSTM) networks in polymeric sciences: A review," *Polymers*, vol. 16, no. 18, p. 2607, 2023. [Online]. Available: https://doi.org/10.3390/polym16182607

[23] S. Al-Selwi, M. Hassan, S. Abdulkadir, and A. Muneer, "LSTM inefficiency in long-term dependencies regression problems," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 30, no. 3, pp. 16–31, 2023.

[24] V. Dogra, S. Verma, P. Chatterjee, J. Shafi, J. Choi, and M. F. Ijaz, "A complete process of text classification system using state-of-the-art NLP models," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, p. 1883698, 2021. [Online]. Available: https://doi.org/10.1155/2022/1883698

[25] Z. Li, A. Hu, X. Wang, J. Hu, and G. Zhang, "Learning to capture dependencies between global features of different convolution layers," *Journal of Visual Communication and Image Representation*, vol. 81, p. 103360, 2021. [Online]. Available: https://doi.org/10.1016/j.jvcir.2021.103360

[26] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, 2021. [Online]. Available: https://doi.org/10.1016/j.aiopen.2022.10.001

[27] X. Mao, S. Chang, J. Shi, F. Li, and R. Shi, "Sentiment-aware word embedding for emotion classification," *Applied Sciences*, vol. 9, no. 7, p. 1334, 2018. [Online]. Available: https://doi.org/10.3390/app9071334

[28] J. Zhou, Z. Ye, S. Zhang, Z. Geng, N. Han, and T. Yang, "Investigating response behaviour through TF-IDF and Word2vec text analysis: A case study of PISA 2012 problem-solving process data," *Heliyon*, vol. 10, no. 16, p. e35945, 2024. [Online]. Available: https://doi.org/10.1016/j.heliyon.2024.e35945

[29] M. Arif, M. Shahiki Tash, A. Jamshidi, F. Ullah, I. Ameer, J. Kalita, A. Gelbukh, and F. Balouchzahi, "Analyzing hope speech from psycholinguistic and emotional perspectives," *Scientific Reports*, vol. 14, no. 1, pp. 1–18, 2024. [Online]. Available: https://doi.org/10.1038/s41598-024-74630-y

[30] V. N. Gudivada, D. L. Rao, and A. R. Gudivada, "Information retrieval: Concepts, models, and systems," *Handbook of Statistics*, vol. 38, pp. 331–401, 2017. [Online]. Available: https://doi.org/10.1016/bs.host.2018.07.009

[31] R. Rana, "Gated recurrent unit (GRU) for emotion classification from noisy speech," *arXiv preprint arXiv:1612.07778*, 2016. [Online]. Available: https://arxiv.org/abs/1612.07778

[32] H. Huang, W. Chen, T. Xie, Y. Wei, Z. Feng, and W. Wu, "The impact of individual behaviors and governmental guidance measures on pandemic-triggered public sentiment based on system dynamics and cross-validation," *International Journal of Environmental Research and Public Health*, vol. 18, no. 8, p. 4245, 2021. [Online]. Available: https://doi.org/10.3390/ijerph18084245

[33] D. B. Olawade, O. Z. Wada, A. Odetayo, A. C. David-Olawade, F. Asaolu, and J. Eberhardt, "Enhancing mental health with artificial intelligence: Current trends and future prospects," *Journal of Medicine, Surgery, and Public Health*, vol. 3, p. 100099, 2024. [Online]. Available: https://doi.org/10.1016/j.glmedi.2024.100099

[34] M. Sykora, S. Elayan, I. R. Hodgkinson, T. W. Jackson, and A. West, "The power of emotions: Leveraging user-generated content for customer experience management," *Journal of Business Research*, vol. 144, pp. 997–1006, 2022. [Online]. Available: https://doi.org/10.1016/j.jbusres.2022.02.048

[35] G. Agag, Y. M. Shehawy, A. Almoraish, R. Eid, H. Chaib Lababdi, T. Gherissi Labben, and S. S. Abdo, "Understanding the relationship between marketing analytics, customer agility, and customer satisfaction: A longitudinal perspective," *Journal of Retailing and Consumer Services*, vol. 77, p. 103663, 2024. [Online]. Available: https://doi.org/10.1016/j.jretconser.2023.103663

[36] X. Yu, S. Wu, W. Chen, and M. Huang, "Sentiment analysis of public opinions on the higher education expansion policy in China," *SAGE Open*, 2021. [Online]. Available: https://doi.org/10.1177/21582440211040778

[37] C. Halkiopoulos and E. Gkintoni, "Leveraging AI in e-learning: Personalized learning and adaptive assessment through cognitive neuropsychology—A systematic analysis," *Electronics*, vol. 13, no. 18, p. 3762, 2023. [Online]. Available: https://doi.org/10.3390/electronics13183762

[38] M. F. Shahzad, S. Xu, W. M. Lim, X. Yang, and Q. R. Khan, "Artificial intelligence and social media on academic performance and mental well-being: Student perceptions of positive impact in the age of smart learning," *Heliyon*, vol. 10, no. 8, p. e29523, 2024. [Online]. Available: https://doi.org/10.1016/j.heliyon.2024.e29523

[39] Y. Cai, X. Li, and J. Li, "Emotion recognition using different sensors, emotion models, methods and datasets: A comprehensive review," *Sensors*, vol. 23, no. 5, p. 2455, 2022. [Online]. Available: https://doi.org/10.3390/s23052455

[40] A. P. Wibawa and F. Kurniawan, "Advancements in natural language processing: Implications, challenges, and future directions," *Telematics and Informatics Reports*, vol. 16, p. 100173, 2024. [Online]. Available: https://doi.org/10.1016/j.teler.2024.100173

[41] D. Mamieva, A. B. Abdusalomov, A. Kutlimuratov, B. Muminov, and T. K. Whangbo, "Multimodal emotion detection via attention-based fusion

of extracted facial and speech features," *Sensors*, vol. 23, no. 12, p. 5475, 2022. [Online]. Available: https://doi.org/10.3390/s23125475

[42] L. Liao, H. Li, W. Shang, and L. Ma, "An empirical study of the impact of hyperparameter tuning and model optimisation on the performance properties of deep neural networks," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 3, 2022. [Online]. doi: 10.1145/3506695

[43] T. Chutia and N. A. Baruah, "A review on emotion detection by using deep learning techniques," *Artificial Intelligence Review*, vol. 57, p. 203, 2024. [Online]. Available: https://doi.org/10.1007/s10462-024-10831-1

[44] M. C. Caschera, P. Grifoni, and F. Ferri, "Emotion classification from speech and text in videos using a multimodal approach," *Multimodal Technologies and Interaction*, vol. 6, no. 4, p. 28, 2022. [Online]. Available: https://doi.org/10.3390/mti6040028

[45] M. Saberi and T. Lilasathapornkit, "Scalability challenges of machine learning models for estimating walking and cycling volumes in large networks," *npj Sustainable Mobility and Transport*, vol. 1, no. 1, pp. 1–16, 2024. [Online]. Available: https://doi.org/10.1038/s44333-024-00009-1

[46] Y. A. Ali, E. M. Awwad, and A. Maarouf, "Hyperparameter search for machine learning algorithms for optimizing the computational complexity," *Processes*, vol. 11, no. 2, p. 349, 2023. [Online]. Available: https://doi.org/10.3390/pr11020349

[47] L. Yang and A. Shami, "On hyperparameter optimisation of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, no. 1, 2020. [Online]. doi: 10.1016/j.neucom.2020.07.061

[48] I. Salehin and D. Kang, "A review on dropout regularization approaches for deep neural networks within the scholarly domain," *Electronics*, vol. 12, no. 14, p. 3106, 2022. [Online]. Available: https://doi.org/10.3390/electronics12143106

[49] N. E. , "Emotions," Kaggle, https://www.kaggle.com/datasets/nelgiriyewithana/emotions?resource=download&select=text.csv (accessed Dec. 20, 2024).