Initial Project Report

Detection of Depression from Textual Data in Social Media Using Machine Learning

Done by

Malavika P (MAC23MCA-2037)

Under the guidance of Prof. Biju Skaria

Introduction

Depression is one of the leading causes of suicide worldwide. However, a significant percentage of depression cases go undiagnosed and untreated. According to the World Health Organization (WHO), depression is the most prevalent mental disorder, affecting over 300 million people globally. It is also responsible for more than two-thirds of suicides annually.

Recent studies have demonstrated that messages posted by individuals with major depressive disorder on social media platforms can be analyzed to predict their likelihood of suffering from depression. Social media platforms, where people freely share their thoughts and feelings, offer a valuable source for monitoring health issues and trends. For instance, posts on platforms like Twitter and Facebook enable researchers to investigate multiple aspects of mental health. Specifically, studies have shown that tweets from individuals with major depressive disorder can be used to predict future episodes of depression.

A recent survey indicated that an increasing number of people, particularly teenagers and young adults, are turning to social media to express their feelings of depression. However, related work in this domain often relies on specific keywords like ‘depression’ and ‘diagnose’ when utilizing the data. In reality, social media users suffering from depression are unlikely to use such words directly due to the social stigma surrounding the condition. As a result, many affected individuals turn to less formal resources, such as social media, to seek support.

This project aims to develop a machine learning model for detecting depression from textual data on social media. By analyzing posts without relying on explicit keywords, the project seeks to identify patterns and signals indicative of depression, providing a tool for early detection and intervention.

Literature Review

Paper 1: Tadesse MM, Lin H, Xu B, Yang L. "Detection of depression-related posts in reddit social media forum." Ieee Access. 2019 Apr 4

The paper titled "Detection of Depression-Related Posts in Reddit Social Media Forum" by Michael M. Tadesse et al. presents a study on identifying depression-related content on Reddit using natural language processing (NLP) and machine learning techniques. The authors address the growing concern of depression as a major contributor to global disability and a significant cause of suicide. They aim to examine Reddit users' posts to detect factors that may reveal depressive attitudes.

The researchers use natural language processing (NLP) and machine learning techniques to train and test their method. They identify words commonly used by people with depression and analyse their connection to mental health. Methods like N-grams (word combinations), LIWC (emotional and psychological content analysis), and LDA (topic identification) help them understand how language on social media can indicate depression.

The study compares single and combined feature sets using various text classification methods like including Logistic Regression, Support Vector Machine, Random Forest, Adaptive Boosting, and Multilayer Perceptron. Results show that proper feature selection and combination lead to higher performance. The best depression detection performance (91% accuracy, 0.93 F1 score) is achieved by combining LIWC, LDA, and bigram features using a Multilayer Perceptron neural network.

This paper helps improve detecting depression on social media, like Reddit, by combining language analysis and machine learning, which could lead to earlier support for those at risk.

Title	Detection of Depression-Related Posts in Reddit Social Media Forum
Area of Work	Mental health detection in social media using NLP and machine learning.
Dataset	The dataset was built by Inna Pirina et al. and consists of a list of depressed and non-depressed users.
Methodology/Strategy	Data pre-processing, Feature extraction, Text classification using various machine learning algorithms.
Algorithm	LR, SVM, RF, AdaBoost, MLP
Result/Precision	91% accuracy and 0.93 F1 score using LIWC+LDA+bigram features with MLP.
Advantages	Combines multiple NLP techniques for improved performance, Examines both single and combined feature sets
Future Proposal	Examine the link between users' personality and depression-related behaviour on social media, Apply the method to other mental health disorders, Test the approach on non-English language data.

Paper 2: Chiong R, Budhi GS, Dhakal S, Chiong F. “A textual-based featuring approach for depression detection using machine learning classifiers and social media texts.” Computers in Biology and Medicine. 2021 Aug 1

The paper "A textual-based featuring approach for depression detection using machine learning classifiers and social media texts" by Raymond Chiong et al. introduces a new way to detect depression by analyzing social media posts with machine learning. The paper focus on the problem of undiagnosed depression and its serious consequences. They propose a method that uses text-based features and different machine learning classifiers to find signs of depression in social media posts, even when specific words like "depression" or "diagnosis" are not used.

The study uses various text preprocessing and feature extraction methods, such as bag-of-words and n-grams. The authors test their approach on multiple datasets from different social media platforms and compare the performance of individual and combined machine learning classifiers. They also tackle the issue of imbalanced datasets with dynamic sampling techniques. The results show that their method effectively detects depression across different social media sources, even when trained on one platform and tested on others. This paper helps advance machine learning, natural language processing, and mental health research by providing a method to detect early signs of depression using social media data.

Title	A textual-based featuring approach for depression detection using machine learning classifiers and social media texts
Area of Work	Depression detection using social media data and machine learning
Dataset	The document references several Kaggle datasets related to depression and suicide analysis: **Shen et al.(2017):** Depression dataset. **Eye BB (2020)**: Focuses on depression analysis. **Tanwar R (2020)**: Based on the diary of a 17-year-old girl, Victoria, who committed suicide due to depression. **Komati N (2020)**: Includes posts from Reddit communities r/SuicideWatch and r/depression. **Virahonda S (2020)**: Consists of comments related to depression and anxiety.
Methodology/Strategy	Text preprocessing, Feature extraction using bag-of-words and n-grams, Training and testing ML models on Twitter datasets, Testing trained models on non-Twitter depression datasets, Experimenting with sampling techniques for imbalanced data.
Algorithm	Single classifiers: LR, SVM, MLP, DT Ensemble classifiers: RF, AdaBoost, Bagging, Gradient Boosting
Result/Precision	Best single classifier: LR (92.61% accuracy on Eye's dataset) Best ensemble: RF (balanced accuracy for depression/non-depression classes) Models performed well even without depression-specific keywords in training data Generalized well to non-Twitter depression datasets
Advantages	Effective depression detection without relying on specific keywords, Generalizable approach that works across different social media platforms
Future Proposal	Incorporate unsupervised learning techniques, Explore multimodal approaches (text + images/video), Investigate more sophisticated NLP techniques, Develop real-time depression monitoring systems, Conduct longitudinal studies to track depression progression.

Paper 3: Islam MR, Kabir MA, Ahmed A, Kamal AR, Wang H, Ulhaq A. "Depression detection from social network data using machine learning techniques." Health information science and systems. 2018 Dec

The paper titled "Depression detection from social network data using machine learning techniques" by Islam et al. presents an innovative approach to detect depression through analysis of social media data, specifically Facebook comments. The authors explore how social networks can be used to monitor and improve mental health and uses machine learning to study the language and emotions in Facebook comments to find signs of depression.

The study utilizes a dataset of 7,145 Facebook comments, categorized into depression-indicative and non-depression-indicative groups. The authors extract psycholinguistic features using the Linguistic Inquiry and Word Count (LIWC) software, focusing on emotional processes, temporal processes, and linguistic style. They then apply and compare four machine learning classifiers: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Trees (DT), and Ensemble methods.

The researchers conduct experiments to evaluate the performance of these classifiers across different feature sets. Their results indicate that the Decision Tree classifier generally outperforms other methods, achieving the highest accuracy in detecting depression-indicative comments. The study also includes a time series analysis, revealing that depression-indicative comments are more prevalent during AM hours.

Overall, this paper contributes to the field of mental health informatics by demonstrating the potential of machine learning techniques in detecting depression through social media data. The findings suggest that automated analysis of social network content could be a valuable tool for early detection and intervention of depression, potentially improving mental health outcomes.

Title	Depression detection from social network data using machine learning techniques
Area of Work	Mental Health, Social Network Analysis
Dataset	Publicly available Facebook data (from bipolar, depression, and anxiety Facebook pages)
Methodology/Strategy	Collected Facebook comments data, Used LIWC software to extract linguistic features, Applied machine learning classifiers
Algorithm	Decision Tree, K-NN, Support Vector Machine, Ensemble Methods
Result/Precision	Decision Tree performed best overall Accuracy between 60-80% for different features
Advantages	Uses real social media data, Examines linguistic, emotional, and temporal features
Future Proposal	Use more types of emotional features, verify techniques on larger datasets, apply more LIWC attributes (used 21 out of 50+)

Summary Table

Paper 1	The paper "Detection of Depression-Related Posts in Reddit Social Media Forum" explores detecting depression on Reddit using natural language processing (NLP) and machine learning. It focuses on analyzing language patterns of depressed users, examining features such as n-grams, LIWC categories, and LDA topic modeling. Various classifiers, including LR, SVM, RF, AdaBoost, and MLP, were tested. The SVM classifier with bigrams achieved 80% accuracy, while combining LIWC, LDA, and bigram features with a multilayer perceptron reached 91% accuracy and a 0.93 F1 score. The study highlights that effective feature selection and integration can enhance depression detection in social media text.
Paper 2	The paper "A Textual-Based Featuring Approach for Depression Detection Using Machine Learning Classifiers and Social Media Texts" by Raymond Chiong et al. presents a method for detecting depression through social media posts without relying on specific keywords. The approach uses textual features extracted via bag-of-words, and n-grams, and evaluates various classifiers, including LR, SVM, MLP, and ensembles. Experiments on Twitter and non-Twitter datasets demonstrate that the method effectively identifies depression with over 90% accuracy. Notably, LR and RF performed best, and dynamic under sampling improved results on imbalanced datasets. The study shows that generalized machine learning techniques can detect depression in social media texts, regardless of keyword usage.
Paper 3	The paper explores detecting depression from Facebook comments using machine learning techniques. It synthesizes emotion detection literature and identifies four feature types: emotional process, temporal process, linguistic style, and a combined set. Experiments were conducted using LIWC software to extract psycholinguistic features, and various classifiers, including Decision Trees, SVM, KNN, and Ensemble methods, were tested. Decision Trees performed best, with the highest recall and F-measure, achieving 60-80% accuracy in detecting depression-indicative comments. The study also includes time series analysis to track depression patterns over different times and months.

Project Proposal

Based on the analysis of the above 3 papers, it is evident that depression is a widespread issue in contemporary society, with social media emerging as an informal platform for individuals to express their feelings. Many individuals suffering from depression tend to express their emotions through social media posts and comments.

In response to this, my proposed system aims to detect depression from textual data posted on social media using machine learning classifiers and feature extraction methods. I have chosen to employ LIWC (Linguistic Inquiry and Word Count), LDA (Latent Dirichlet Allocation), N-gram, and Bag-of-Words (BoW) methods to extract features for detecting depression in text, regardless of specific keywords such as ‘depressed’ or ‘not depressed’. The system synthesizes emotion detection literature and identifies 4 types of features: emotional process, linguistic style, temporal process and a combined set.

The project intends to utilize Multi-Layer Perceptron (MLP) and Random Forest (RF) for ensemble methods, as well as Decision Trees (DT) to achieve the most accurate results. Additionally, the system will include a feature that suggests remedies for users identified as having depression.

The expected output of the system is to analyse text or any social media comment or post. If the post indicates any tone of sadness or contains depressive content, the model will determine that the user is experiencing depression, and vice versa. Furthermore, the system will provide suggested remedies for users identified as depressed.

To facilitate user interaction, I plan to implement a web interface using Flask. This interface will allow users to input text to be evaluated for depression. The output will indicate whether the text suggests the user is experiencing depression, and if so, will provide suggested remedies in an additional text field.

By utilizing advanced machine learning techniques and comprehensive feature extraction methods, the system aims to provide an accurate and practical tool for detecting depression in social media text and offering supportive suggestions.

Background and Motivation:

Depression is becoming more common in today's society, and many people use social media to share their feelings and experiences. Recent studies show that we can use what people post online to identify signs of depression, which can help provide early support and intervention.

The motivation for this project comes from key findings and approaches in recent studies:

- Using NLP and Machine Learning: A Reddit study (paper 1) showed that natural language processing (NLP) combined with machine learning can effectively detect depression-related content with high accuracy (91%) and a strong F1 score (0.93).

- Generalized Feature Extraction: A study by Raymond Chiong (paper 2) and others found that depression can be detected in social media texts without specific keywords. Techniques like bag-of-words and n-grams achieved over 90% accuracy, showing these methods are robust.

- Diverse Feature Sets: Studies emphasized the importance of diverse features. The Reddit study used n-grams, LIWC categories, and LDA topic modeling. The Facebook study (paper 3) identified emotional process, temporal process, linguistic style, and combined features as useful. This varied approach forms the basis of our system's comprehensive feature set.

- Classifier Performance: Different classifiers performed well in different studies. SVM was effective in the Reddit study, while Logistic Regression and Random Forest excelled in the Twitter study. Decision Trees worked well in the Facebook study. This diversity in classifier performance motivates us to use multiple classifiers like Multi-Layer Perceptron, Random Forest, and Decision Trees.

- Handling Imbalanced Data: The Twitter study (paper 2) showed that dynamic under-sampling improves results on imbalanced datasets, a common challenge in detecting depression.

- Temporal Analysis: The Facebook study used time series analysis to track depression patterns over time, suggesting the value of incorporating temporal features in detection.

- Cross-Platform Applicability: Success across Reddit, Twitter, and Facebook indicates the potential for a general system to detect depression on various platforms.

- Need for Support: These studies focus on detection, but there's a need for follow-up support. Our system includes a feature to suggest remedies for potentially depressed users.

- Accessibility: The high prevalence of depression and widespread social media use highlight the need for accessible, scalable screening tools. We chose to use a web interface with Flask to make the tool easily accessible.

Our proposed system aims to be comprehensive, accurate, and user-friendly for detecting depression in social media text. It builds on successful approaches in recent research and focuses on practical application and user support. By integrating advanced feature extraction methods, multiple classifiers, and a supportive interface, we hope to contribute to early depression detection and intervention in the digital age.

Methodology

Data Collection

- Collect data from existing datasets that resides in websites like Kaggle, UCI repository, GitHub etc.

Data Preprocessing

- Remove URLs, mentions, hashtags, emojis, and special characters from the text
- Split the text into individual words or phrases.
- Convert all text to lowercase and remove common words that don't add much meaning.

Feature Extraction

- LIWC (Linguistic Inquiry and Word Count): Extract psychological and linguistic features.
- N-gram: Generate unigrams, bigrams, and trigrams for contextual information.
- Bag-of-Words (BoW): Create a sparse matrix representing word frequency.

Feature Engineering

- Emotional Process Features: Analyse emotional expressions using sentiment analysis tools.
- Linguistic Style Features: Examine writing style, including personal pronouns and sentence complexity.
- Combined Feature Set: Integrate all extracted features for model training.

Model Development

- Multilayer Perceptron (MLP): Develop a neural network model to classify text and predict remedies.
- Random Forest (RF): Implement an ensemble learning method for enhanced prediction accuracy.
- Decision Tree (DT): Build a simple decision tree model for feature importance and interpretability.

Model Training and Evaluation

- Split the dataset into training and testing sets and perform the evaluation

Model Deployment

- Use Flask to create a user-friendly web interface.
- Develop an input form for users to submit text for evaluation.
- Display analysis results, indicating whether the text suggests depression and providing suggested remedies.
- Deploy the Flask application on a web server.

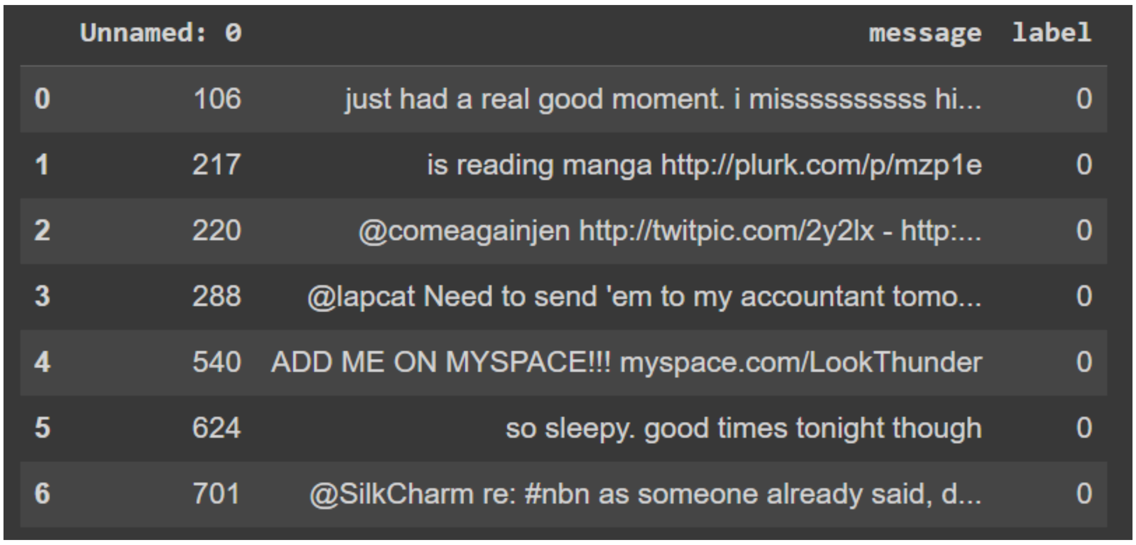
Dataset

To develop the model outlined in the proposed system, we will utilize the dataset “Depression analysis” by BB Eye as mentioned in paper 2 to train the model effectively for detecting depression from social media textual data, such as posts or comments, even in the absence of explicit keywords like "depressed" or "not depressed." Additionally, the issue of imbalanced datasets will be addressed using a dynamic sampling technique. The dataset to be employed are as follows:

- **Eye BB (2020):** This dataset focuses on depression analysis. Contains 10314 rows. 3 columns including text, id and label which is a class identifier. Class identifier identifies the text as depressive or non-depressive. 0 indicates non-depressive and 1 indicates depressive texts.

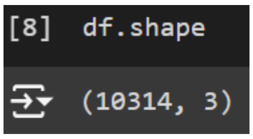
This dataset will be used to train and test the model to effectively detect depression-related content.

Dataset Link: https://www.kaggle.com/datasets/bababullseye/depression-analysis



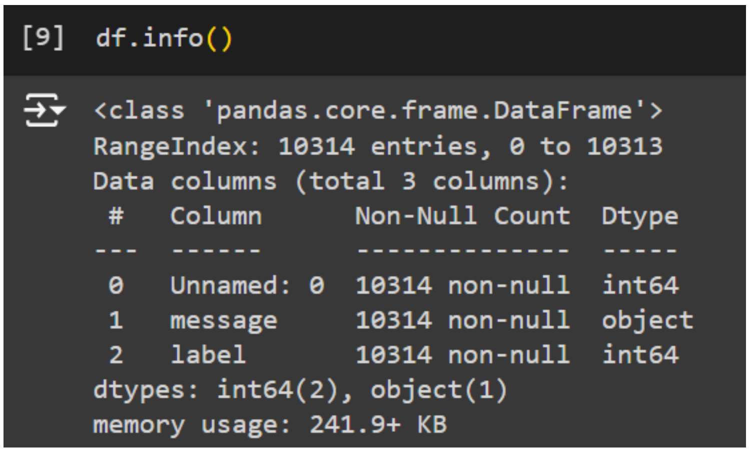
	Unnamed: 0	message	label
0	106	just had a real good moment. i missssssssss hi...	0
1	217	is reading manga http://plurk.com/p/mzp1e	0
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0
3	288	@lapcat Need to send 'em to my accountant tomo...	0
4	540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0
5	624	so sleepy. good times tonight though	0
6	701	@SilkCharm re: #nbn as someone already said, d...	0

Exploratory Analysis



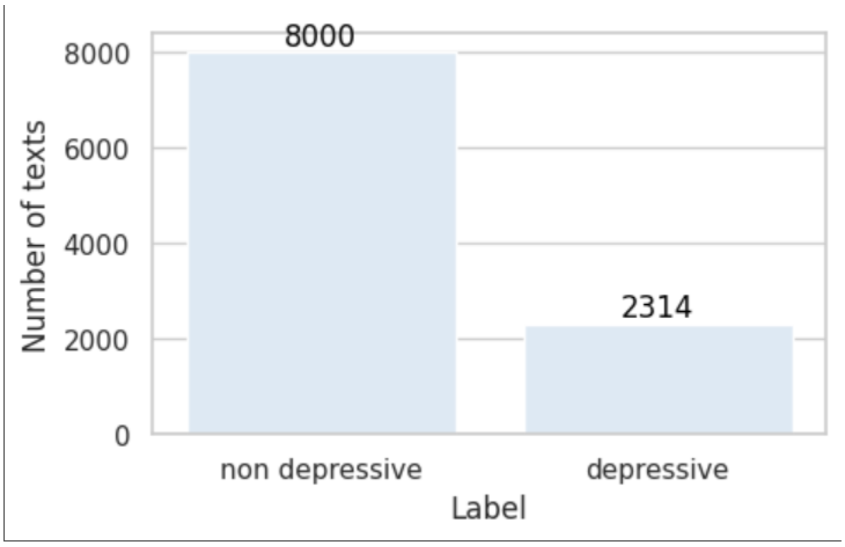
```
[8]  df.shape

(10314, 3)
```

Dataset has 10314 rows and 3 columns.



```
[9]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10314 entries, 0 to 10313
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  10314 non-null  int64 
 1   message     10314 non-null  object
 2   label       10314 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 241.9+ KB
```

Dataset contains 1 non-numerical and 2 numerical attributes out of which unnamed attribute is irrelevant and label attribute is remapped into depressive and non-depressive where 0 indicates non-depressive and 1 indicates depressive.

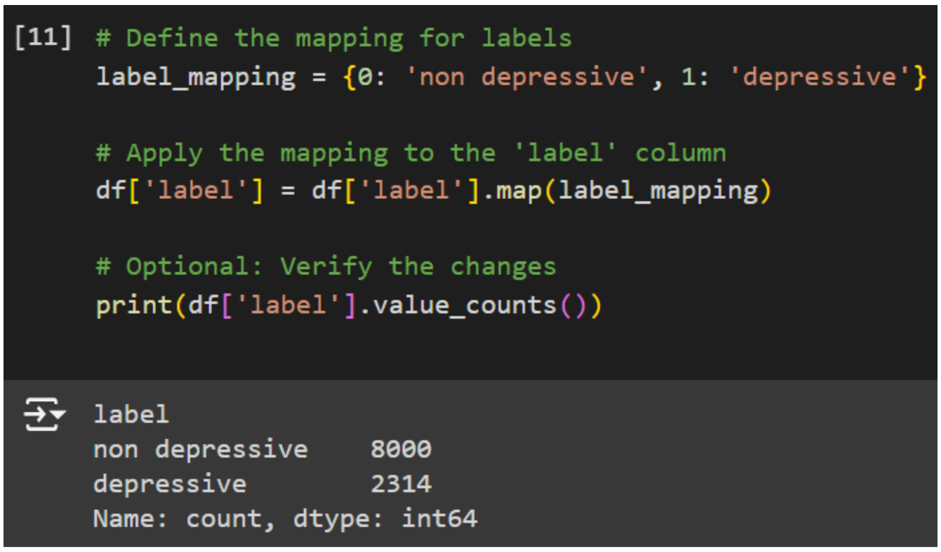


Dataset contains 8000 non-depressive texts and 2314 depressive texts.

Data Preprocessing

Remapping Label Class

The proposed system will classify the text into depressive and non-depressive texts. The label class is remapped into Depressive and Non depressive.

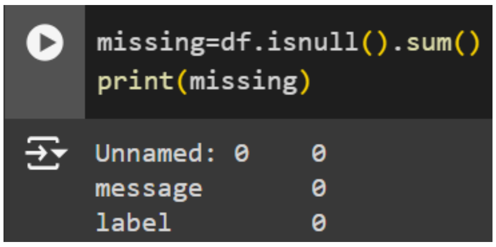


```
[11] # Define the mapping for labels
     label_mapping = {0: 'non depressive', 1: 'depressive'}

     # Apply the mapping to the 'label' column
     df['label'] = df['label'].map(label_mapping)

     # Optional: Verify the changes
     print(df['label'].value_counts())
```

```
label
non depressive    8000
depressive        2314
Name: count, dtype: int64
```

Missing Values

There are no missing values in the dataset.

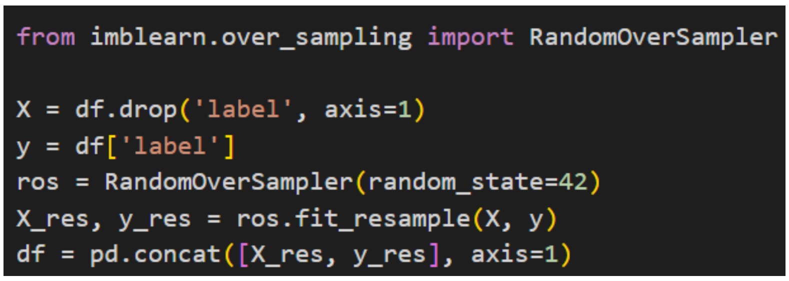


```
missing=df.isnull().sum()
print(missing)
```

```
Unnamed: 0    0
message       0
label         0
```

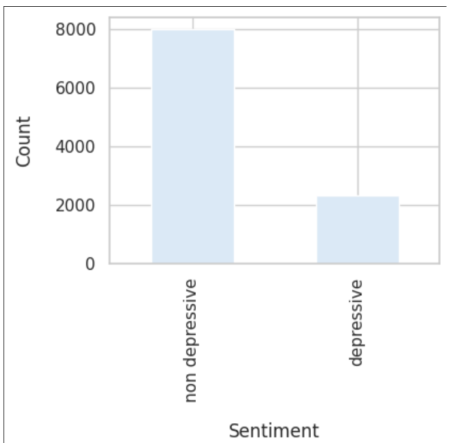
Handling Class imbalance

The dataset is significantly imbalanced, with a large number of non-depressive texts and a comparatively smaller number of depressive texts. This issue has been addressed using a resampling technique.

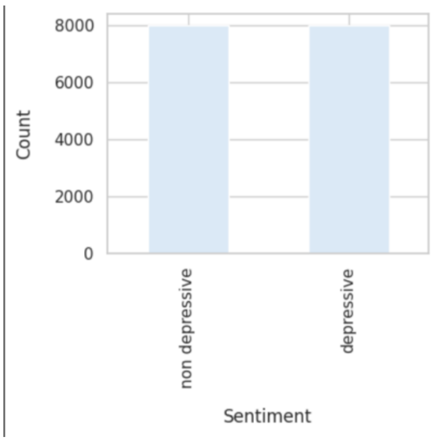


```python
from imblearn.over_sampling import RandomOverSampler

X = df.drop('label', axis=1)
y = df['label']
ros = RandomOverSampler(random_state=42)
X_res, y_res = ros.fit_resample(X, y)
df = pd.concat([X_res, y_res], axis=1)
```

Imbalanced Label Class



Balanced Label Class



Text Preprocessing

The preprocessing tasks aim to clean and standardize the Twitter data, making it more suitable for subsequent analysis or modelling. The provided code successfully performs these tasks, and the processed tweets can be used for various natural language processing tasks, sentiment analysis, or machine learning applications.

NLTK is a powerful library for natural language processing in Python. It provides various tools and resources for tasks such as tokenization, stemming, and stopword removal. In this project, NLTK is employed to handle English stopwords and perform stemming. The re library is used for pattern matching and manipulation of strings. It plays a crucial role in the removal of mentions, hyperlinks, and other specific patterns from Twitter data.

- Removal of Extra Spaces, Mentions, and Hyperlinks: Regular expressions are used to remove extra spaces, mentions (starting with '@'), and hyperlinks from the tweets.

- Removal of Punctuation, Numbers, and Capitalization: Punctuation and numbers are removed, and the messages are converted to lowercase. This step ensures uniformity and reduces noise in the text.

- Tokenization: The lowercase, punctuation-stripped tweets are tokenized into lists of words using the split() function.

- Stemming: The Porter stemming algorithm is applied to each tokenized tweet, reducing words to their stems.

- Removal of Stopwords: Stopwords, including custom Twitter-specific terms, are removed from the tokenized and stemmed tweets.

- Dataset Update: The processed messages are added as a new column ('processed_messages') to the original dataset.

Working of Feature Extraction Methods

LIWC

LIWC analyses text to understand the emotions, thoughts, and social aspects behind the words. It works by comparing the words in a piece of text to a special dictionary that groups words into categories like "happy," "sad," or "reasoning." By counting how often words from each category appear, LIWC can give insights into the emotional tone and reasoning patterns reflected in the text. It's used in research, psychology, and even marketing to better understand how people express themselves.

Pseudocode

```
1. Load LIWC Dictionary
   - Define a dictionary where keys are categories (e.g., 'positive', 'negative')
   - Each key maps to a list of words associated with that category

2. Load Text
   - Obtain the text that needs to be analysed

3. Preprocess Text
   - Convert text to lowercase
   - Remove punctuation and special characters
   - Tokenize the text into words

4. Count Word Occurrences
   - Create a dictionary to count the frequency of each word in the text

5. Analyse Text
   - Initialize a dictionary to store category percentages
   - Calculate the total number of words in the text
   - For each category in the LIWC dictionary:
     - Count how many words from the text match words in the current category
     - Compute the percentage of these words relative to the total number of words
     - Store the percentage in the results dictionary

6. Output Results
   - Print or return the percentage of words for each category
```

N-Gram

An n-gram is a sequence of words (or sometimes characters) taken from a text. The "n" in n-gram refers to the number of words in the sequence. For example, in the phrase "the cat sat," a 2-gram (bigram) would be "the cat" and "cat sat," while a 3-gram (trigram) would be "the cat sat." N-grams are used in text analysis to understand patterns, such as which words commonly follow each other, and are often used in tasks like text prediction and language modelling.

Pseudocode

```
1. Define Function generate_ngrams(text, n):
   - Convert text to lowercase
   - Remove punctuation and special characters
   - Tokenize text into words

2. Initialize an empty list to store n-grams

3. For i from 0 to (length of words - n):
   - Extract the sequence of n words starting at index i
   - Append this sequence to the list of n-grams

4. Return the list of n-grams

5. Example usage:
   - text = "The cat sat on the mat"
   - n = 2 (for bigrams)
   - Call generate_ngrams(text, n)
   - Output: ["the cat", "cat sat", "sat on", "on the", "the mat"]
```

BOW

Bag of Words (BoW) is a simple method used to represent text data. It turns text into a list of word frequencies or occurrences, ignoring grammar and word order. For example, if you have the sentence "The cat sat on the mat," BoW would count how often each word appears and create a list or vector with these counts. This helps in analyzing and comparing texts based on the words they contain.

Pseudocode

```
1. Define Function generate_bow(text):
   - Convert text to lowercase
   - Remove punctuation and special characters
   - Tokenize text into words

2. Initialize an empty dictionary to store word counts

3. For each word in the list of words:
   - If the word is not in the dictionary:
      - Add the word to the dictionary with a count of 1
   - If the word is already in the dictionary:
      - Increment the word's count by 1

4. Return the dictionary of word counts

5. Example usage:
   - text = "The cat sat on the mat"
   - Call generate_bow(text)
   - Output: {'the': 2, 'cat': 1, 'sat': 1, 'on': 1, 'mat': 1}
```

Working of Algorithms

Algorithms used in ‘Detection of Depression from Textual Data in social media Using Machine Learning’ are Random Forest, Decision Tree and Multilayer Perceptron.

Random Forest

Random Forest is a supervised learning technique used for both Classification and Regression problems. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Instead of relying on one decision tree, the random forest takes the output from each tree and based on the majority votes of outputs, and it takes the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Working

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Pseudocode

```
1. Input:
   - Training dataset with N samples and M features.
   - Number of trees to create: T.
   - Number of features to randomly select at each split: F.
2. Initialize an empty list called 'forest' to store the trees.
3. For each tree (from 1 to T):
   a. Create a random sample of the training data (some samples may repeat).
   b. Build a decision tree:
      i. At each decision point in the tree:
         - Randomly pick F features from the M features.
         - Find the best feature and split point among the F features to split the node.
         - Split the node into two child nodes based on the selected split point.
      ii. Keep splitting until the tree is fully grown or meets some stopping condition (like a max
depth or minimum samples).
```

```
  c. Add this decision tree to the 'forest'.
4. To make a prediction for a new sample x:
  a. Let each tree in the forest make a prediction for the new sample x.
  b. Aggregate the predictions:
    - For classification: Use majority voting to determine the final prediction.
    - For regression: Take the average of all the tree predictions.
```

Decision Tree:

Decision trees are a type of machine-learning algorithm that can be used for both classification and regression tasks. They work by learning simple decision rules inferred from the data features. These rules can then be used to predict the value of the target variable for new data samples.

Working

Step 1: Start with the entire dataset at the root node.

Step 2: Select the best feature to split the data at the root node.

Step 3: Split the dataset into subsets based on the chosen feature and split point.

Step 4: Repeat Steps 2 & 3 for each subset to build the tree.

Step 5: Assign a final prediction to each leaf node.

Step 6: For new data points, pass them down the tree to reach a leaf node and output the prediction.

Pseudocode

```
1. Input:
  - Training dataset with N samples and M features.
  - Stopping criteria, such as maximum depth or minimum number of samples per leaf.
2. Initialize the root of the tree with the entire dataset.
3. For each node in the tree:
  a. If all samples at the node belong to the same class (classification) or the maximum depth is reached, make the node a leaf node and assign the class label or average value.
  b. Else:
    i. Calculate the best feature and split point based on a criterion (like Gini Impurity, Information Gain for classification, or Variance Reduction for regression).
    ii. Split the dataset into two subsets based on the best feature and split point.
    iii. Create child nodes for each subset.
    iv. Recursively apply steps 3a and 3b to the child nodes.
```

4. Output: A fully grown decision tree.

5. To make a prediction for a new sample x:

   a. Start at the root node.

   b. Traverse the tree by following the decision rules at each internal node.

   c. Continue until a leaf node is reached.

   d. The prediction is the value or class label assigned to the leaf node.

Multilayer Perceptron

A Multilayer Perceptron (MLP) is a type of artificial neural network used for supervised learning tasks, such as classification and regression. It consists of an input layer, one or more hidden layers, and an output layer, where each layer is made up of neurons. These neurons are fully connected between layers and use activation functions like ReLU or Sigmoid to learn complex, non-linear relationships in the data. The network is trained using backpropagation, where the weights are adjusted iteratively to minimize the error between predicted and actual outputs. MLPs are versatile and can be applied to a wide range of problems.

Step 1: Input Layer

- The input features are fed into the input layer, with each feature represented by a neuron.

Step 2: Forward Propagation

- **Hidden Layers**: Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.
- This process repeats through all hidden layers.

Step 3: Output Layer

- The final layer computes the output, applying an activation function if needed (e.g., Softmax for classification).

Step 4: Loss Calculation

- The difference between the predicted and actual output is calculated using a loss function.

Step 5: Backpropagation

- The error is backpropagated to update the weights using gradient descent.

Step 6: Training

- Repeat forward propagation, loss calculation, and backpropagation for several iterations to minimize the loss.

Step 7: Prediction

- After training, the MLP makes predictions on new data using forward propagation.

Pseudocode

```
1. Input:
   - Training dataset with N samples and M features.
   - Number of hidden layers: L.
   - Number of neurons in each hidden layer.
   - Activation function (e.g., ReLU, Sigmoid).
   - Learning rate ($\alpha$).
   - Number of epochs (iterations).
2. Initialize weights and biases for all layers.
3. For each epoch:
   a. For each sample in the training dataset:
      i. Forward Propagation:
         - For each layer from 1 to L:
            1. Compute the weighted sum: $z = (weights * input) + bias$.
            2. Apply activation function: $a = activation(z)$.
            3. Set the output of the current layer as input for the next layer.
         - Compute the final output in the output layer.
      ii. Compute the loss using the loss function (e.g., Mean Squared Error, Cross-Entropy).
      iii. Backward Propagation:
         - Calculate the gradient of the loss with respect to each weight and bias.
         - Update weights and biases using gradient descent:
           weight = weight - $\alpha$ * (gradient of loss with respect to weight)
           bias = bias - $\alpha$ * (gradient of loss with respect to bias)
4. After training, use the learned weights and biases to make predictions on new data by performing forward propagation.
5. Output:
   - Trained model that can make predictions on new data.
```

Packages and Functions

Pandas

- read_csv()
- head()
- shape()
- info()
- drop()
- value_counts()
- isnull()

Matplotlib

- subplots(figsize=(5, 3))
- set_xlabel()
- set_ylabel()
- annotate()
- show()

Seaborn

- set(style="whitegrid")
- set_palette("Blues")
- barplot(x=class_labels, y=class_counts, ax=ax)

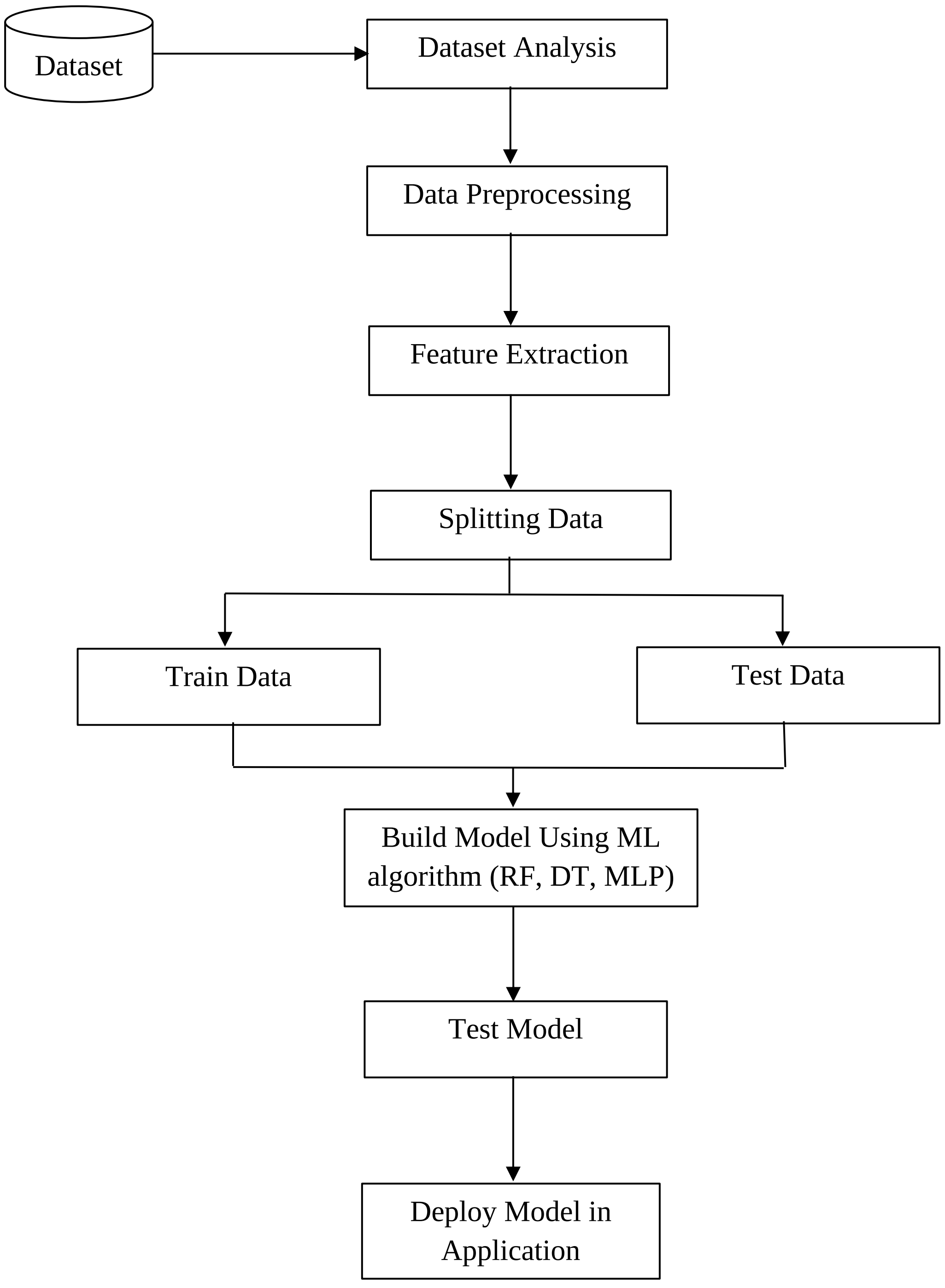
NLTK (Natural Language Toolkit)

- corpus.stopwords.words()
- PorterStemmer()

re (Regular Expressions)

- re.compile():
- str.replace()

Project Pipeline



TIMELINE

- Submission of project synopsis with Journal Papers - 22.07.2024
- Project proposal approval - 26.07.2024
- Presenting project proposal before the Approval Committee - 29.07.2024 & 30.07.2024
- Initial report submission - 12.08.2024
- Analysis and design report submission - 16.08.2024
- Verification of the report and PPT by the guide - 19.08.2024
- First project presentation - 21.08.2024 & 23.08.2024
- Sprint Release I - 30.08.2024
- Sprint Release II - 26.09.2024
- Interim project presentation - 30.09.2024 & 01.10.2024
- Sprint Release III - 18.10.2024
- Project execution, submission of project report and PPT before the guide - 24.10.2024
- Submission of the project report to the guide - 28.10.2024
- Final project presentation - 28.10.2024 & 29.10.2024
- Submission of project report after corrections - 01.11.2024