Human Stress Prediction Project Report

Introduction to Data Mining and Machine Learning

FAMNIT, Koper

Marija Ćetković

July 26, 2024

1 Introduction

This Data Mining and Machine Learning project aims to tackle the challenge of identifying psychological stress from textual data. As such it falls into the category of Natural Language Processing, and the project aims to apply NLP techniques to handle the problem. The project is based on a Kaggle dataset 'Human Stress Prediction' derived from mental health-related subreddits and it explores the efficiency of different machine learning models in classifying the posts as either stressed or non-stressed.

Although the task of stress detection is complex due to the variability in language and expression across different individuals, leveraging machine learning to analyze social media posts can be of great value for assessing both individual users' and platforms' overall mental health and well-being.

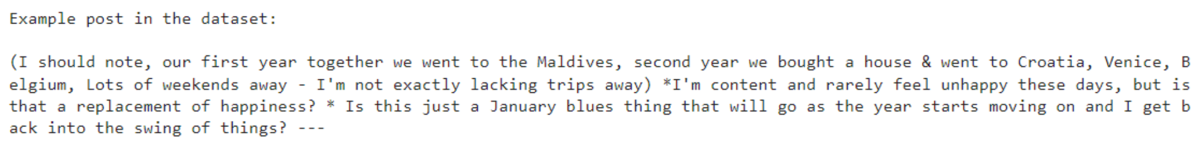
2 Data Description

The dataset used in this project is the "Human Stress Prediction" dataset, which contains social media posts labeled as either stressed (1) or non-stressed (0). The dataset has the following columns:

- **subreddit**: The subreddit where the post was made.
- **post_id**: The unique identifier for the post.
- **sentence_range**: The range of sentences in the post.
- **text**: The content of the post.
- **label**: The stress label (1 for stressed, 0 for non-stressed).
- **confidence**: The confidence of the label.
- **social_timestamp**: The timestamp of the post.

3 Exploratory Data Analysis (EDA)

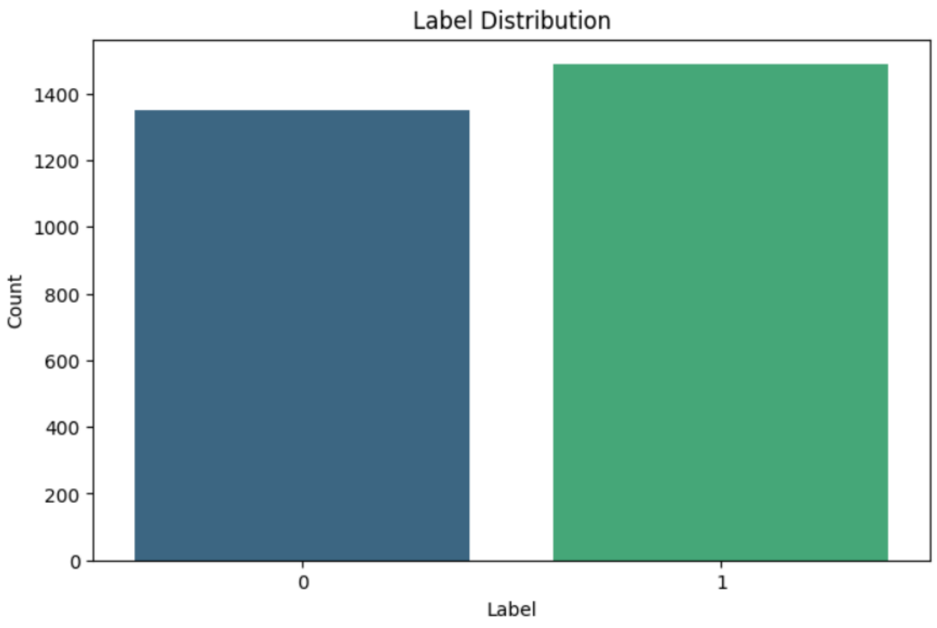
3.1 Example post in the dataset



```
Example post in the dataset:

(I should note, our first year together we went to the Maldives, second year we bought a house & went to Croatia, Venice, B
elgium, Lots of weekends away - I'm not exactly lacking trips away) *I'm content and rarely feel unhappy these days, but is
that a replacement of happiness? * Is this just a January blues thing that will go as the year starts moving on and I get b
ack into the swing of things? ---
```

3.2 Label distribution

We can notice that the dataset is balanced with respect to the labels we want to predict.



3.3 Basic post statistics

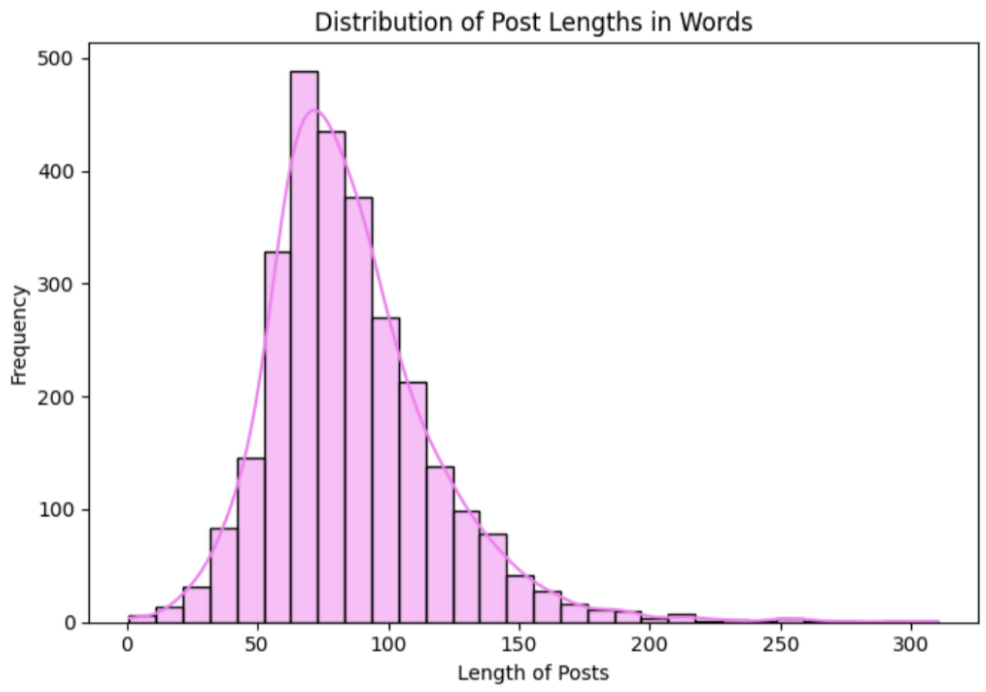
The basic statistics for post lengths in words reveal the following:

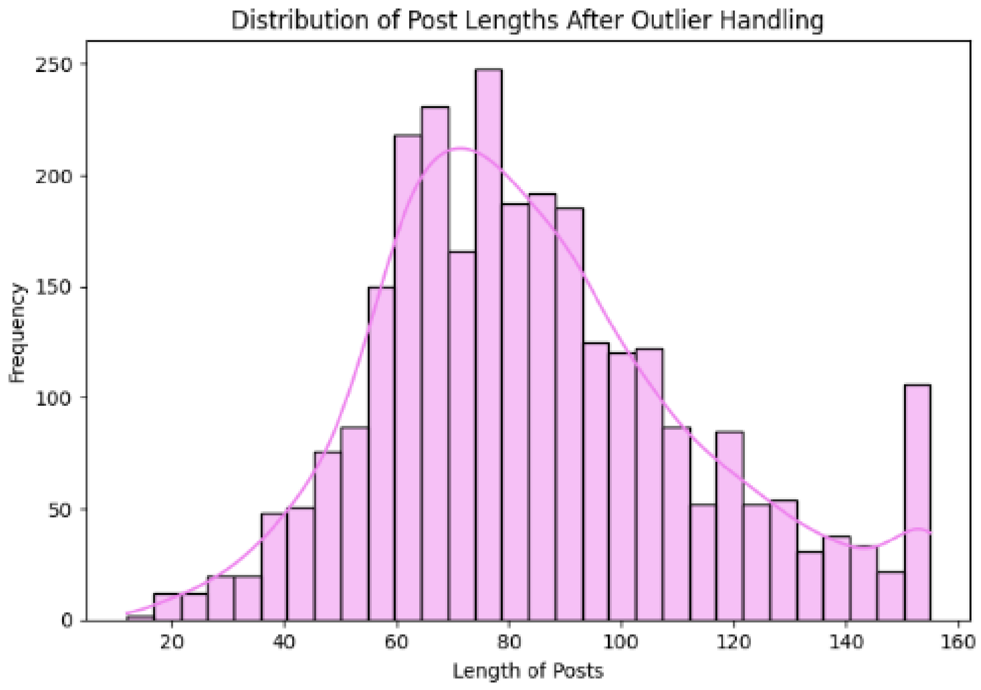
- **Count**: 2,838 posts were analyzed.
- **Mean**: The average post length is approximately 86 words.
- **Standard Deviation**: There is a moderate variation in post length, with a standard deviation of about 32 words.
- **Minimum**: The shortest post contains just 1 word.
- **25th Percentile**: 25% of posts have 65 words or fewer.
- **Median (50th Percentile)**: The median post length is 80 words, indicating that half of the posts are shorter than this.
- **75th Percentile**: 75% of posts have 101 words or fewer.
- **Maximum**: The longest post has 310 words.

These statistics highlight the diversity in post lengths (maximum with respect to median). The removal of outliers was experimented with, but revealed not much difference with accuracy of the models' prediction, possibly because of the size of the dataset.

3.4 Outlier Handling

We identified 6 outliers below the lower bound (Q1 - 1.5 * IQR) and 87 above the upper bound (Q3 + 1.5 * IQR), with the bounds being 11 and 155 in words, respectively. Removing or truncating (upper outliers) these outliers improved the distribution of post lengths in terms of words. However, this adjustment did not significantly affect the accuracy, so the dataset with removed/truncated outliers was not used, but we can present the distribution charts.





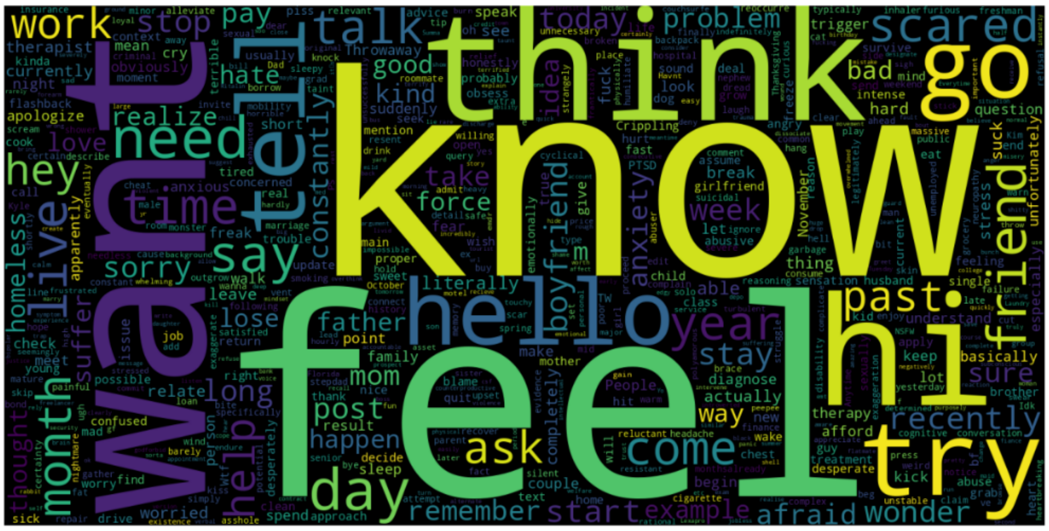
4 Text Preprocessing

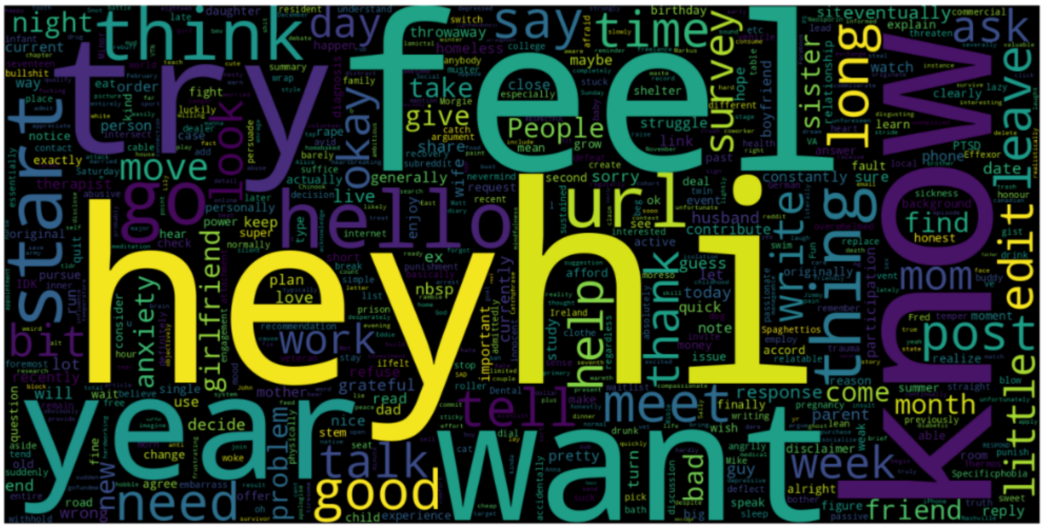
Text preprocessing is an important step in NLP, that enhances the quality of raw text data. In this project, a few techniques were applied. Tokenization, which breaks text into smaller units (splitting sentences into words); lemmatization, which reduces words to their base forms (*saying* → *say*); and the removal of stopwords (”and”, ”the”, ”is” that do not provide actual meaning to the data) and punctuation characters. This cleaning phase is done with the goal of improving the accuracy of models, with preserving the core meaning of the data (posts). In this case, the spaCy library was utilized.

```
import spacy

nlp = spacy.load("en_core_web_lg")

def preprocess(post):
    doc = nlp(post)

    clean_tokens = []
    for token in doc:
        if token.is_punct or token.is_stop or not token.text.isalpha():
            continue
        clean_tokens.append(token.lemma_)

    return clean_tokens
```

5 Data Visualization

5.1 Word Clouds





5.2 Top Words Analysis

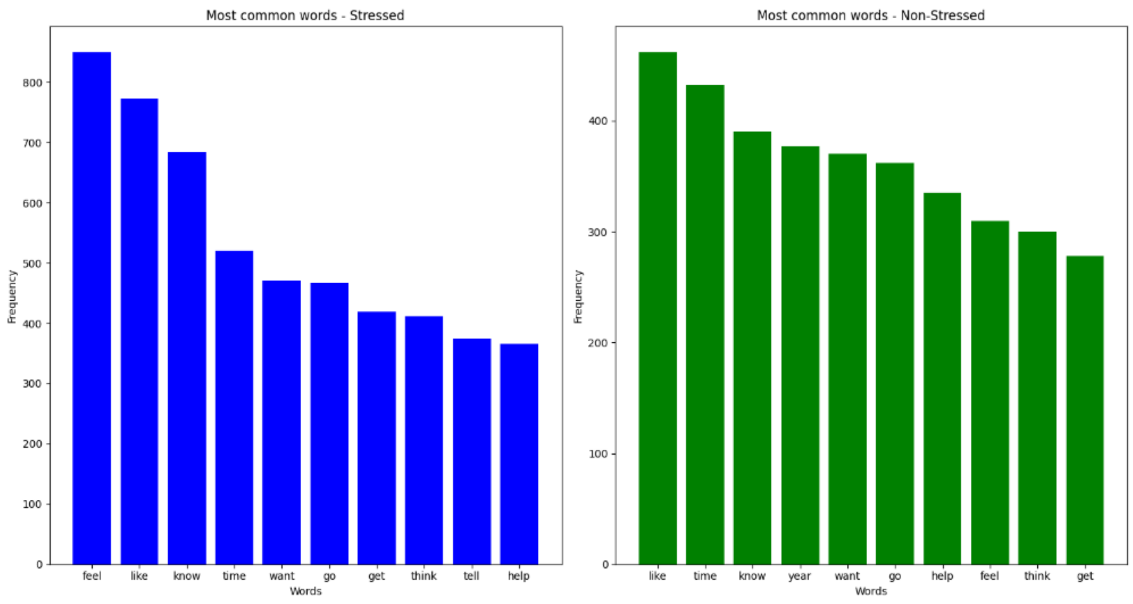


Figure 1: Enter Caption

6 Preparing Data for Model Training

The processed text data needs to be transformed into numerical representations to feed into the algorithms. Three representations were tested: **Bag-of-Words (BoW)**, **Bag-of-N-grams (BoN)** and **Word2Vec**.

6.1 Bag-of-Words

The **Bag-of-Words (BoW)** model creates a vocabulary of all unique words present in the training dataset. Each document is then represented as a vector, where each element corresponds to the frequency or presence of a word in the document. This method treats each word independently, ignoring the order in which words appear.

```
# Convert tokenized data to strings
X_train_str = [' '.join(doc) for doc in X_train]
X_test_str = [' '.join(doc) for doc in X_test]

# Create the TF-IDF vectorizer
tf_vectorizer = TfidfVectorizer(min_df=0, max_df=1, use_idf=True, ngram_range=(1, 1))

# Tokenize and build vocabulary on the training data
bow_train = tf_vectorizer.fit_transform(X_train_str)

# Encode the test data using the same vocabulary
bow_test = tf_vectorizer.transform(X_test_str)
```

6.2 Bag-of-N-grams

The **Bag-of-N-grams (BoN)** model considers sequences of words, called n-grams, instead of individual words. This approach captures some of the contextual information by taking more than an individual word as input.

```
# Create the transform
tf_vectorizer2 = TfidfVectorizer(analyzer='word', ngram_range=(1, 3))

# Tokenize and build vocabulary on the training data
bon_train = tf_vectorizer2.fit_transform(X_train_str)

# Encode the test data using the same vocabulary
bon_test = tf_vectorizer2.transform(X_test_str)
```

6.3 Word2Vec

The **Word2Vec** model represents words in a continuous vector space where semantically similar words are mapped to nearby points (words with similar appear close to each other in the N-dimensional space). Thus it aids understanding word meanings and contexts.

In this implementation, we use the pre-trained GloVe Twitter word vectors with 25 dimensions, which is loaded using the Gensim library's API. GloVe (Global Vectors for Word Representation) is an unsupervised learning algorithm for obtaining vector representations for words.Training uses global word-word co-occurrence statistics from a corpus, resulting in word vectors with useful linear relationships.

```
wv = api.load("glove-twitter-25")
```

In this process, each word in the post is checked against the GloVe word embedding model. If the word is present in the model vocabulary, its corresponding vector representation is retrieved using `wv_model.get_vector(word)`, which results in a list of word vectors representing the entire post. This technique is suitable for the LSTM model used, as it preserves post context and supports the sequential nature of model's data prediction.

```
for word in post:
    if word in wv_model:
        vectorized_post.append(wv_model.get_vector(word))
```

Alternatively, we could use mean vector approach, which involves averaging the word vectors for all words in a given post to obtain a single vector representation for the entire post. This method ensures that each post is represented by a fixed-length vector, regardless of the number of words it contains. This approach is used for feeding the data into ML models that require fixed-size input (Naive Bayes, Linear Regression, SVM).

```
for tokenized_post in X_train:
    if tokenized_post:
        wv_train_data.append(wv.get_mean_vector(tokenized_post))
```

The Word2Vec model provides a dense vector (in comparison to sparse matrices in previous models) representation for each post, capturing the semantic information and relationships between words more effectively than the previous ones.

7 Models

7.1 Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is a probabilistic classifier based on Bayes' theorem with independence assumptions between features. In this project, `GaussianNB` from `sklearn.naive_bayes` was used.

```
gnb_bow = GaussianNB()
gnb_bow.fit(bow_train.toarray(), y_train)
y_pred_bow = gnb_bow.predict(bow_test.toarray())
```

7.2 Support Vector Machine

Support Vector Machines are classifiers that find the best dividing line (or hyperplane) to separate different groups of data points. Here the `SVC` from `sklearn.svm` is used. SVMs are effective in high-dimensional spaces and can handle the sparse nature of text data.

```
svm_bow = SVC(kernel='linear', random_state=4)
svm_bow.fit(bow_train, y_train)
y_pred_bow = svm_bow.predict(bow_test)

svm_w2v = SVC(kernel='rbf', random_state=4)
```

The kernel for Word2Vec was switched from linear to radial basis function, which produced better results for this type of representation, as it was more adequate for non-linear data pattern recognition.

7.3 Logistic Regression

Logistic Regression is a linear model used for binary classification tasks. It predicts the probability of a sample belonging to a particular class, by applying the logistic function on the input features. In this project, `LogisticRegression` from `sklearn.linear_model` was used.

```
logreg_bow = LogisticRegression(max_iter=1000, random_state=4)
logreg_bow.fit(bow_train, y_train)
y_pred_bow = logreg_bow.predict(bow_test)
```

7.4 Long Short-Term Memory

Long Short-Term Memory networks are a type of Recurrent Neural Network capable of learning long-term dependencies. This project uses `Sequential`, `LSTM`, and `Dense` layers from `tensorflow.keras`.

The LSTM model, with its ability to capture temporal patterns, is well-suited for handling sequential data like text. Training involved using word embeddings from `glove-twitter-25`, optimizing with the Adam optimizer. (BOW and BON representations were not included for simplicity.)

The process for preparing data and training an LSTM model for binary classification involves the function `create_sequences_and_labels` that converts tokenized text data into sequences of word vectors using the Word2Vec model. Each sequence consists of word vectors of a fixed length. Here, sequences of length 20 with word vectors of size 25 are prepared for both training and test datasets.

The model features an LSTM layer with 128 units, Dropout for regularization, and a Dense output layer with a sigmoid activation function for binary classification. L2 regularization is applied to both the LSTM and Dense layers to prevent overfitting.

The model is trained using the Adam optimizer for 20 epochs and a batch size of 128.

```
seq_length = 20
vector_size = 25
```

```
X_train_lstm, y_train_lstm = create_sequences_and_labels(X_train,
y_train, wv, seq_length, vector_size)
X_test_lstm, y_test_lstm = create_sequences_and_labels(X_test,
y_test, wv, seq_length, vector_size)
model = Sequential()
model.add(LSTM(128, input_shape=(seq_length, 25), return_sequences=False,
kernel_regularizer='l2'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid', kernel_regularizer='l2'))

adam = Adam(learning_rate=3e-4)
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])

model.fit(X_train_lstm, y_train_lstm, epochs=20, batch_size=128,
validation_data=(X_test_lstm, y_test_lstm))
loss, accuracy = model.evaluate(X_test_lstm, y_test_lstm)
```

8 Results

Model	**BOW Accuracy**	**BON Accuracy**	**Word2Vec Accuracy**
Gaussian Naive Bayes	0.5591	0.6314	0.6931
Logistic Regression	0.5608	0.6984	0.7002
SVM	0.5626	0.7231	0.7002
LSTM			0.6752

9 Conclusion

The project demonstrates the application of NLP and machine learning techniques to predict human stress levels from social media posts. Future work could explore more advanced models, larger datasets and hopefully utilizing the confidence level of the posts.