Human Stress Detection

Read more →

KEANECLEMENT, HENDRA PUTRA, KEVIN MAXWELL

INTRODUCTION

01. In the digital age, the volume of textual data created and distributed by individuals has grown at an exponential rate. This data includes diverse sources like social media, emails, text messages, and many other platforms. By analyzing this textual data, we can extract valuable information about an individual's emotional state, as well as their stress levels.

02. Stress is a typical reaction that people experience when confronted with daily pressures and challenges. When stress persists, it can have detrimental effects on an individual's physical and mental wellbeing.

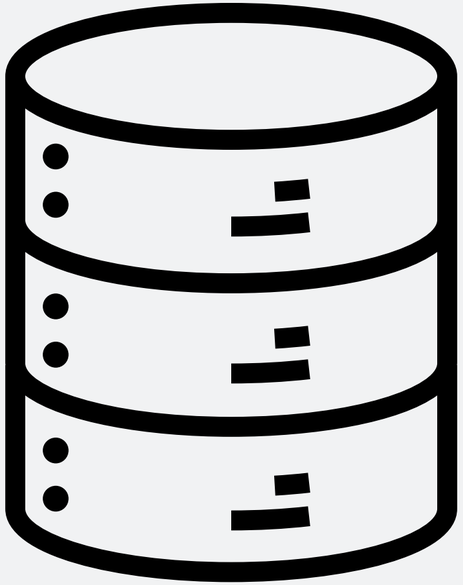
03. The objective of this presentation is to present the idea and techniques of human stress recognition through text analysis

PURPOSE

to utilize advanced machine learning algorithms to analyze text inputs from users and predict their stress levels. By processing and interpreting textual data, such as journal entries, social media posts, and other written communications, the app aims to provide accurate and timely insights into a user's stress state.

01. **Early Detection:** To identify early signs of stress from users' text inputs before they manifest into more serious mental or physical health issues.

02. **Continuous Monitoring:** To provide continuous monitoring of stress levels by analyzing ongoing text inputs, ensuring users are aware of their stress states in real-time.



DATASET

The Kaggle dataset that we used consist of 2839 data and 7 column/features which includes information that can be used to forecast the stress levels of people. The dataset contains demographic data and a variety of physiological signs that can be utilized to build stress detection models. It seeks to support the development of machine learning models that use these signals to forecast stress.

METHOD

Text Cleaning

Cleans the text by removing unnecessary characters like punctuation and URLs. It ensures the text is clean and ready for analysis.

Stopword Removal

Removes common words (stopwords) from the text that don't carry much meaning. This helps focus on important words.

Stemming Method

Reduces words to their base form. For example, 'running' and 'runs' become 'run'. It helps to treat similar words as the same.

Word Tokenization

the process of breaking down a piece of text into individual words or tokens.It involves splitting the text into smaller units based on specific criteria, usually spaces or punctuation marks.

METHOD

TF-IDF Vectorization

This method converts a collection of raw documents (text) into a matrix of TF-IDF features. TF-IDF stands for Term Frequency-Inverse Document Frequency.

Model Training

This method is responsible for training a Machine Learning model, making predictions, and evaluating its performance. It accepts a model object (such as Logistic Regression, Naive Bayes, etc.) and a title (for identification) as parameters

Cross-Validation

a statistical technique used to assess the performance of a predictive model. It is commonly employed in machine learning to evaluate how well a model will generalize to an independent dataset.

Prediction Model

prediction model utilizes a trained model to analyze new text and determine whether it exhibits signs of stress. By applying the model's learned patterns and features to the input text, it generates a prediction indicating the likelihood of stress presence

IMPORT LIBRARIES

```
import seaborn as sns
import numpy as np
import pandas as pd
import nltk
import re
from urllib.parse import urlparse
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('wordnet2022')
nltk.download('punkt')
nltk.download('stopwords')
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score
import string
```

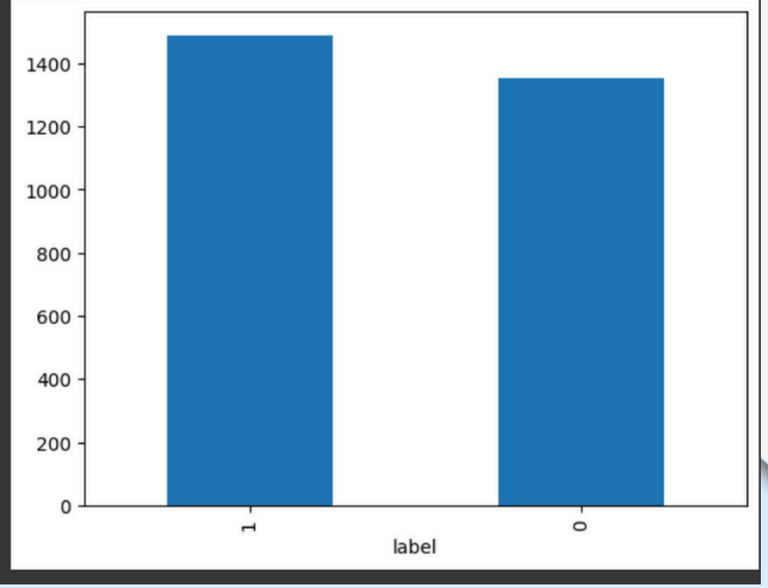
LOAD DATASET

```
[ ] df = pd.read_csv('Stress.csv')
    df.head()
```

	subreddit	post_id	sentence_range	text	label	confidence	social_timestamp
0	ptsd	8601tu	(15, 20)	He said he had not felt that way before, sugge...	1	0.8	1521614353
1	assistance	8lbrx9	(0, 5)	Hey there r/assistance, Not sure if this is th...	0	1.0	1527009817
2	ptsd	9ch1zh	(15, 20)	My mom then hit me with the newspaper and it s...	1	0.8	1535935605
3	relationships	7rorpp	[5, 10]	until i met my new boyfriend, he is amazing, h...	1	0.6	1516429555
4	survivorsofabuse	9p2gbc	[0, 5]	October is Domestic Violence Awareness Month a...	1	0.8	1539809005

```
df.label.value_counts().plot.bar()
```

```
<Axes: xlabel='label'>
```



VISUALISATION

PRE-PROCESSING

```
## clean_text
def clean_text(text):
    text = str(text).lower()
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
## remove_stopwords
stop_words = stopwords.words('english')
more_stopwords = ['u', 'im', 'c']
stop_words = stop_words + more_stopwords
def remove_stopwords(text):
    words = text.split(' ')
    words = [word for word in words if word not in stop_words]
    text = ' '.join(words)
    return text
```

```
##  stem_text
import nltk
stemmer = PorterStemmer()
def stem_text(text):
    text = ' '.join(stemmer.stem(word) for word in text.split(' '))
    return text
## preprocess_data
def preprocess_data(text):
    # Clean puntuation, urls, and so on
    text = clean_text(text)
    # Remove stopwords
    text = ' '.join(word for word in text.split(' ') if word not in stop_words)
    # Stemm all the words in the sentence
    text = ' '.join(stemmer.stem(word) for word in text.split(' '))
    return text
# Function to remove ticks from the middle of words
def remove_ticks(text):
    text = text.replace("“", "")
    text = text.replace("”", "")
    return text.replace("’", "")
```

CLEANED DATASET

```
print(df['clean_text'])
```

```
0        said felt way sugget go rest trigger ahead you...
1        hey rassist sure right place post goe  current...
2        mom hit newspap shock would know dont like pla...
3        met new boyfriend amaz kind sweet good student...
4        octob domest violenc awar month domest violenc...
                              ...
2833      week ago preciou ignor  jan  happi year preci...
2834     dont abil cope anymor tri lot thing trigger sh...
2835     case first time read post look peopl complet o...
2836     find normal good relationship main problem see...
2837     talk mom morn said sister trauma wor mine didn...
Name: clean_text, Length: 2838, dtype: object
```

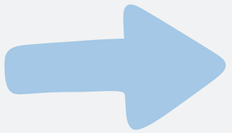
TRAIN MODEL USING TF IDF VECTORIZER

```
def modelling(model, title = "Default"):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print('Accuracy', title, ':', accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

```
[ ]  tfidf = TfidfVectorizer()
     tfidf_df = tfidf.fit_transform(feature['clean_text'])
     tfidf_df.toarray()
     tfidf_df = pd.DataFrame(tfidf_df.toarray(), columns=tfidf.get_feature_names_out())
     X = tfidf_df
     y = feature['label']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

NAIVE BAYES AND KNN METHOD

```python
# Naive Bayes
gnb = GaussianNB()
modelling(gnb, "Gaussian Naive Bayes")
mnb = MultinomialNB()
modelling(mnb, "Multinomial Naive Bayes")
# KNN
knn = KNeighborsClassifier(n_neighbors=5)
modelling(knn, "KNN")
```



```
Accuracy Gaussian Naive Bayes : 0.5880281690140845
              precision    recall  f1-score   support

           0       0.60      0.46      0.52       414
           1       0.58      0.71      0.64       438

    accuracy                           0.59       852
   macro avg       0.59      0.58      0.58       852
weighted avg       0.59      0.59      0.58       852
```

```
Accuracy Multinomial Naive Bayes : 0.67018779342723
              precision    recall  f1-score   support

           0       0.88      0.37      0.52       414
           1       0.62      0.95      0.75       438

    accuracy                           0.67       852
   macro avg       0.75      0.66      0.64       852
weighted avg       0.74      0.67      0.64       852
```

```
Accuracy KNN : 0.6314553990610329
              precision    recall  f1-score   support

           0       0.65      0.52      0.58       414
           1       0.62      0.74      0.67       438

    accuracy                           0.63       852
   macro avg       0.64      0.63      0.63       852
weighted avg       0.63      0.63      0.63       852
```

DECISION TREES, RANDOM FOREST AND LOGISTIC REGRESSION

```python
# Decission trees
dtc = DecisionTreeClassifier()
modelling(dtc, "Decission trees")
# Random Forest
rf = RandomForestClassifier(n_estimators=1
modelling(rf, "Random Forest")
# Logistic Regression
lg = LogisticRegression()
modelling(lg, "Logistic Regression")
```

```
Accuracy Decission trees : 0.602112676056338
              precision    recall  f1-score   support

           0       0.60      0.55      0.57       414
           1       0.60      0.66      0.63       438

    accuracy                           0.60       852
   macro avg       0.60      0.60      0.60       852
weighted avg       0.60      0.60      0.60       852
```

```
Accuracy Random Forest : 0.7230046948356808
              precision    recall  f1-score   support

           0       0.79      0.59      0.67       414
           1       0.69      0.85      0.76       438

    accuracy                           0.72       852
   macro avg       0.74      0.72      0.72       852
weighted avg       0.73      0.72      0.72       852
```

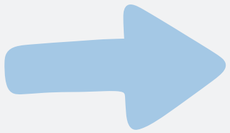
```
Accuracy Logistic Regression : 0.7476525821596244
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       414
           1       0.74      0.79      0.76       438

    accuracy                           0.75       852
   macro avg       0.75      0.75      0.75       852
weighted avg       0.75      0.75      0.75       852
```

TRAIN MODEL USING COUNT VECTORIZER

```
[ ]  cv = CountVectorizer()
     cv_df = cv.fit_transform(feature['clean_text'])
     cv_df.toarray()
     cv_df = pd.DataFrame(cv_df.toarray(),columns=cv.get_feature_names_out())
     X = cv_df
     y = feature['label']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

NAIVE BAYES AND KNN METHOD

```python
# Naive Bayes
gnb = GaussianNB()
modelling(gnb, "Gaussian Naive Bayes")
mnb = MultinomialNB()
modelling(mnb, "Multinomial Naive Bayes")
# KNN
knn = KNeighborsClassifier(n_neighbors=5)
modelling(knn, "KNN")
```



```
Accuracy Gaussian Naive Bayes : 0.6044600938967136
              precision    recall  f1-score   support

           0       0.64      0.43      0.51       414
           1       0.59      0.77      0.67       438

    accuracy                           0.60       852
   macro avg       0.61      0.60      0.59       852
weighted avg       0.61      0.60      0.59       852
```

```
Accuracy Multinomial Naive Bayes : 0.7335680751173709
              precision    recall  f1-score   support

           0       0.80      0.60      0.69       414
           1       0.70      0.86      0.77       438

    accuracy                           0.73       852
   macro avg       0.75      0.73      0.73       852
weighted avg       0.75      0.73      0.73       852
```

```
Accuracy KNN : 0.5434272300469484
              precision    recall  f1-score   support

           0       0.52      0.94      0.67       414
           1       0.75      0.17      0.28       438

    accuracy                           0.54       852
   macro avg       0.63      0.55      0.47       852
weighted avg       0.64      0.54      0.47       852
```

DECISION TREES, RANDOM FOREST AND LOGISTIC REGRESSION

```python
# Decission trees
dtc = DecisionTreeClassifier()
modelling(dtc, "Decission trees")
# Random Forest
rf = RandomForestClassifier(n_estimators=1
modelling(rf, "Random Forest")
# Logistic Regression
lg = LogisticRegression()
modelling(lg, "Logistic Regression")
```

```
Accuracy Decission trees : 0.6115023474178404
              precision    recall  f1-score   support

           0       0.60      0.58      0.59       414
           1       0.62      0.64      0.63       438

    accuracy                           0.61       852
   macro avg       0.61      0.61      0.61       852
weighted avg       0.61      0.61      0.61       852
```

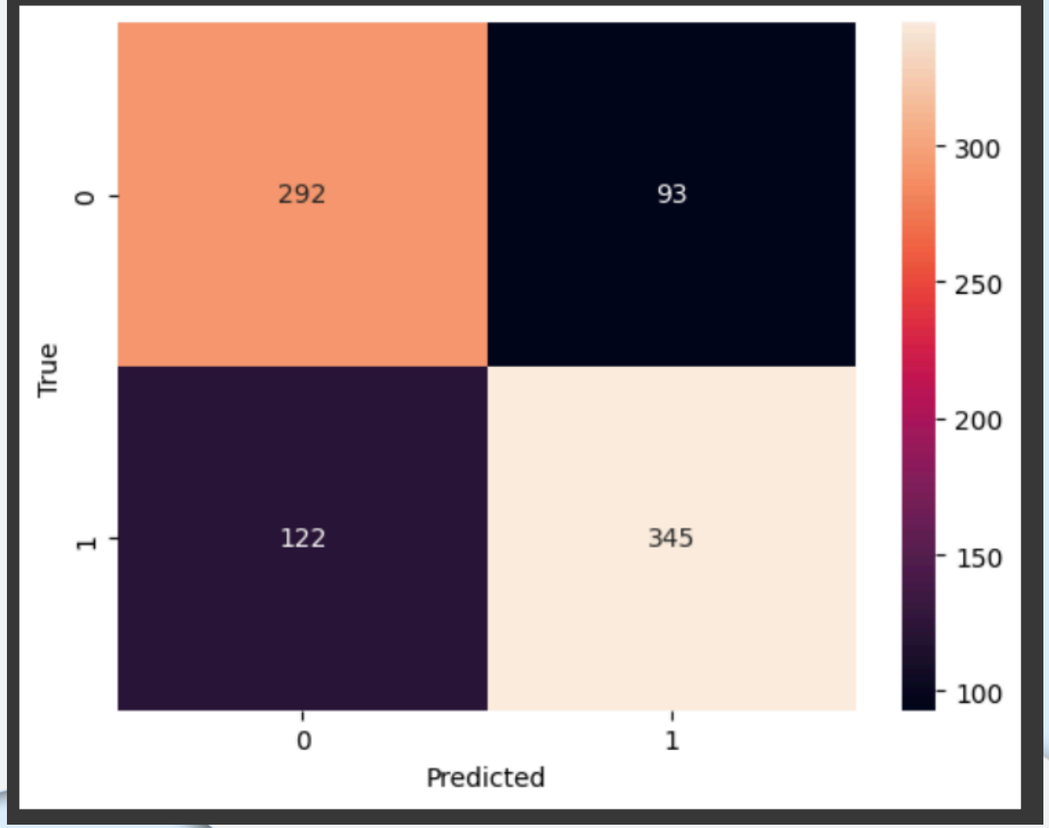
```
Accuracy Random Forest : 0.7065727699530516
              precision    recall  f1-score   support

           0       0.76      0.57      0.65       414
           1       0.67      0.83      0.74       438

    accuracy                           0.71       852
   macro avg       0.72      0.70      0.70       852
weighted avg       0.72      0.71      0.70       852
```

```
Accuracy Logistic Regression : 0.7147887323943662
              precision    recall  f1-score   support

           0       0.72      0.68      0.70       414
           1       0.71      0.75      0.73       438

    accuracy                           0.71       852
   macro avg       0.72      0.71      0.71       852
weighted avg       0.71      0.71      0.71       852
```

CROSS VALIDATION USING K-FOLD

```
model_lg = lg.fit(X_train, y_train)
# Cross Validation
kf = KFold(n_splits=5, shuffle=True, random_state=1)
scores = cross_val_score(lg, X, y, cv=kf)
print("Cross Validation Score using K-Fold:\n", scores)
print("")
y_pred = model_lg.predict(X_test)
cm = confusion_matrix(y_pred, y_test)
sns.heatmap(cm,annot=True)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

```
Cross Validation Score using K-Fold:
 [0.75176056 0.73239437 0.74647887 0.72486772 0.77425044]
```



PREDICTION FUNCTION

```python
def predictor(text):
    process_text = remove_ticks(text)
    process_text = clean_text(process_text)
    process_text = remove_stopwords(process_text)
    process_text = stem_text(process_text)
    process_text = preprocess_data(process_text)
    embedded_words = tfidf.transform([process_text])
    res = model_lg.predict(embedded_words)
    if res[0] == 1:
        res = "this person is in stress"
    else:
        res = "this person is not in stress"
    return res
```

DEMO

```
text1 = "I can't keep up with everything. There's just too much to do!"
text2 = "I haven't been sleeping well at all; I feel exhausted."
text3 = "I'm so overwhelmed right now. I don't know where to start."
text4 = "I'm looking forward to the weekend and spending time with friends."
text5 = "I've got a good handle on my tasks; everything is under control."
```

```
print(predictor(text1))
print(predictor(text2))
print(predictor(text3))
print(predictor(text4))
print(predictor(text5))
```

```
this person is in stress
this person is in stress
this person is in stress
this person is not in stress
this person is not in stress
```

CONCLUSION

The code is designed to build a system that detects whether someone is stressed based on their written text. It starts by loading a dataset containing labeled text (stressed or not). Then, the text goes through cleaning and processing steps. After that, several machine learning models are trained using this processed text. Each model's performance is evaluated to see how well it predicts stress. This system aims to provide insights into someone's stress levels through their written words.

01. **TF/IDF** was the best feature extraction method compared to CountVectorizer

02. **Logistic Regression** Achieved the best performing model, which will be used to predict stress level in new text entries

03. This system **helps predict someone's stress level** by their written words

THANK YOU VERY MUCH!