In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import string

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import re

import gensim

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import tensorflow as tf
```

In [2]:
```
df = pd.read_csv('Combined Data.csv')
df.head()
```

Out[2]:

	Unnamed: 0	statement	status
0	0	oh my gosh	Anxiety
1	1	trouble sleeping, confused mind, restless hear...	Anxiety
2	2	All wrong, back off dear, forward doubt. Stay ...	Anxiety
3	3	I've shifted my focus to something else but I'...	Anxiety
4	4	I'm restless and restless, it's been a month n...	Anxiety

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53043 entries, 0 to 53042
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  53043 non-null  int64 
 1   statement   52681 non-null  object
 2   status      53043 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.2+ MB
```

In [4]:
```
obj = [col for col in df.columns if df[col].nunique() <= 10]

for col in obj:
  print(f"Column: {col}")
  print(f"Number of Unique Values (nunique): {df[col].nunique()}")
  print(f"Unique Values: {df[col].unique()}")
  print("Value Counts:")
```

```
    print(df[col].value_counts())
    print("-" * 50)
```

```
Column: status
Number of Unique Values (nunique): 7
Unique Values: ['Anxiety' 'Normal' 'Depression' 'Suicidal' 'Stress' 'Bipolar'
 'Personality disorder']
Value Counts:
status
Normal                  16351
Depression              15404
Suicidal                10653
Anxiety                  3888
Bipolar                  2877
Stress                   2669
Personality disorder     1201
Name: count, dtype: int64
--------------------------------------------------
```

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
0
```

In [6]:
```
df = df.dropna(subset=['statement'])
```

In [7]:
```
df = df[['statement', 'status']]
df.head()
```

Out[7]:

	statement	status
0	oh my gosh	Anxiety
1	trouble sleeping, confused mind, restless hear...	Anxiety
2	All wrong, back off dear, forward doubt. Stay ...	Anxiety
3	I've shifted my focus to something else but I'...	Anxiety
4	I'm restless and restless, it's been a month n...	Anxiety

In [8]:
```
def clean_text(text):
    if not isinstance(text, str):
        return text
    text = re.sub(r'http[s]?://\S+|www\.\S+', '', text)
    text = re.sub(r"\[.*?\]\(.*?\)", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r'\b\d{1,3}[MF]\b', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'\w*\d\w*', '', text)
    text = text.lower()
    text = text.strip()
    return text

df['statement'] = df['statement'].apply(clean_text)
df.head()
```

Out[8]:

	statement	status
0	oh my gosh	Anxiety
1	trouble sleeping confused mind restless heart ...	Anxiety
2	all wrong back off dear forward doubt stay in ...	Anxiety
3	ive shifted my focus to something else but im ...	Anxiety
4	im restless and restless its been a month now ...	Anxiety

In [9]:
```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

def remove_stopwords_lemmatize(text):
    stop_words = set(stopwords.words('english'))
    stop_words.update(["im", "ive", "dont", "cant", "wont", "youre", "didnt", "does
    lemmatizer = WordNetLemmatizer()
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words
    return " ".join(words)

df['statement'] = df['statement'].apply(remove_stopwords_lemmatize)
df.head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Meera\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Meera\AppData\Roaming\nltk_data...
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Meera\AppData\Roaming\nltk_data...
```

Out[9]:

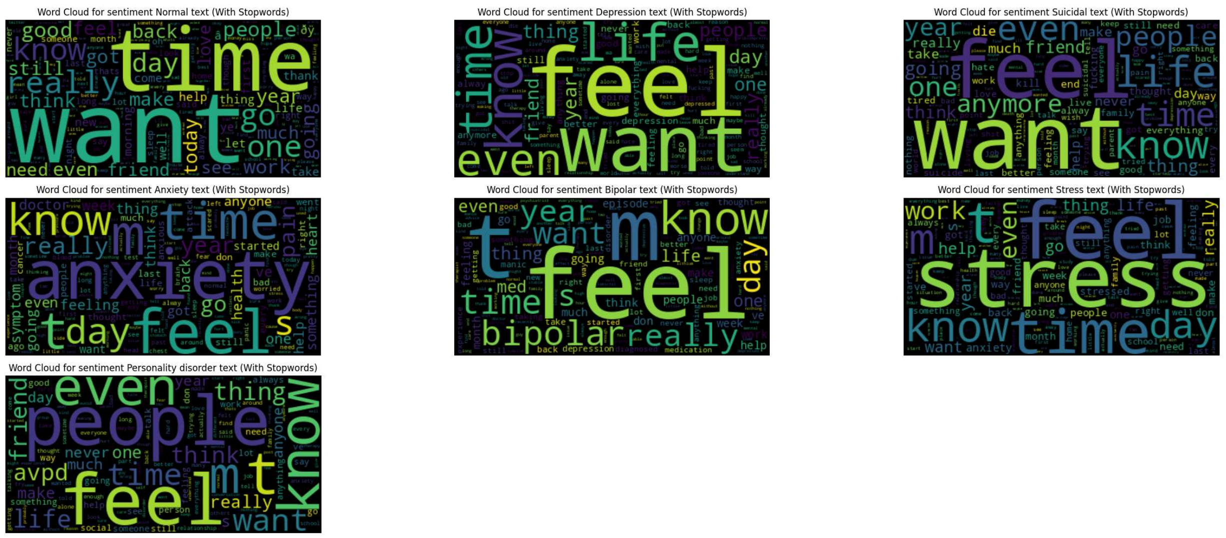
	statement	status
0	oh gosh	Anxiety
1	trouble sleeping confused mind restless heart ...	Anxiety
2	wrong back dear forward doubt stay restless re...	Anxiety
3	shifted focus something else still worried	Anxiety
4	restless restless month boy mean	Anxiety

In [10]:
```
fig, axes = plt.subplots(3, 3, figsize=(25, 10))
axes = axes.flatten()
status = ['Normal', 'Depression', 'Suicidal', 'Anxiety', 'Bipolar', 'Stress', 'Pers

for i, stat in enumerate(status):
    if stat in df['status'].unique():
        statement = ' '.join(df[df['status'] == stat]['statement'])
        if statement:
            wordcloud = WordCloud(max_words=2000, width=400, height=200, collocatio
            axes[i].imshow(wordcloud, interpolation='bilinear')
            axes[i].axis("off")
            axes[i].set_title(f"Word Cloud for sentiment {stat} text (With Stopword
    else:
        axes[i].axis("off")
        axes[i].set_title(f"No data for label {stat}")

for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



In [11]:
```
def mapping(status):
    if status == 'Normal':
        return 0
    if status == 'Depression':
        return 1
    if status == 'Suicidal':
        return 2
    if status == 'Anxiety':
        return 3
    if status == 'Bipolar':
        return 4
    if status == 'Stress':
        return 5
    if status == 'Personality disorder':
        return 6

df['status'] = df['status'].apply(mapping)
df.head()
```

Out[11]:

	statement	status
0	oh gosh	3
1	trouble sleeping confused mind restless heart ...	3
2	wrong back dear forward doubt stay restless re...	3
3	shifted focus something else still worried	3
4	restless restless month boy mean	3

In [12]:

```
words = []
for i in df['statement'].values:
    words.append(i.split())

EMBEDDING_DIM = 200

w2v_model = gensim.models.Word2Vec(sentences = words, vector_size = EMBEDDING_DIM,
                                   window = 5, min_count = 3)
w2v_model.train(words, total_examples = len(words), epochs = 10)
```

Out[12]:

```
(23707026, 26797280)
```

In [13]:

```
tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words = 50000)
tokenizer.fit_on_texts(words)
tokenized_train = tokenizer.texts_to_sequences(words)

X = tf.keras.preprocessing.sequence.pad_sequences(tokenized_train, maxlen=100)
```

In [14]:

```
vocab_size = len(tokenizer.word_index) + 1
```

In [15]:

```
def get_weight_matrix(model, tokenizer):
    vocab = tokenizer.word_index
    vocab_size = len(vocab) + 1
    weight_matrix = np.zeros((vocab_size, EMBEDDING_DIM))

    for word, i in vocab.items():
        if word in model.wv:
            weight_matrix[i] = model.wv[word]
    return weight_matrix
```

In [16]:
```
embedding_vectors = get_weight_matrix(w2v_model, tokenizer)
```

In [17]:
```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Embedding(vocab_size, output_dim=EMBEDDING_DIM, weights=[
model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64, recurrent_dr
model.add(tf.keras.layers.Dense(7, activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['a
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	14,129,800
bidirectional (Bidirectional)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)

Total params: 14,129,800 (53.90 MB)

Trainable params: 14,129,800 (53.90 MB)

Non-trainable params: 0 (0.00 B)

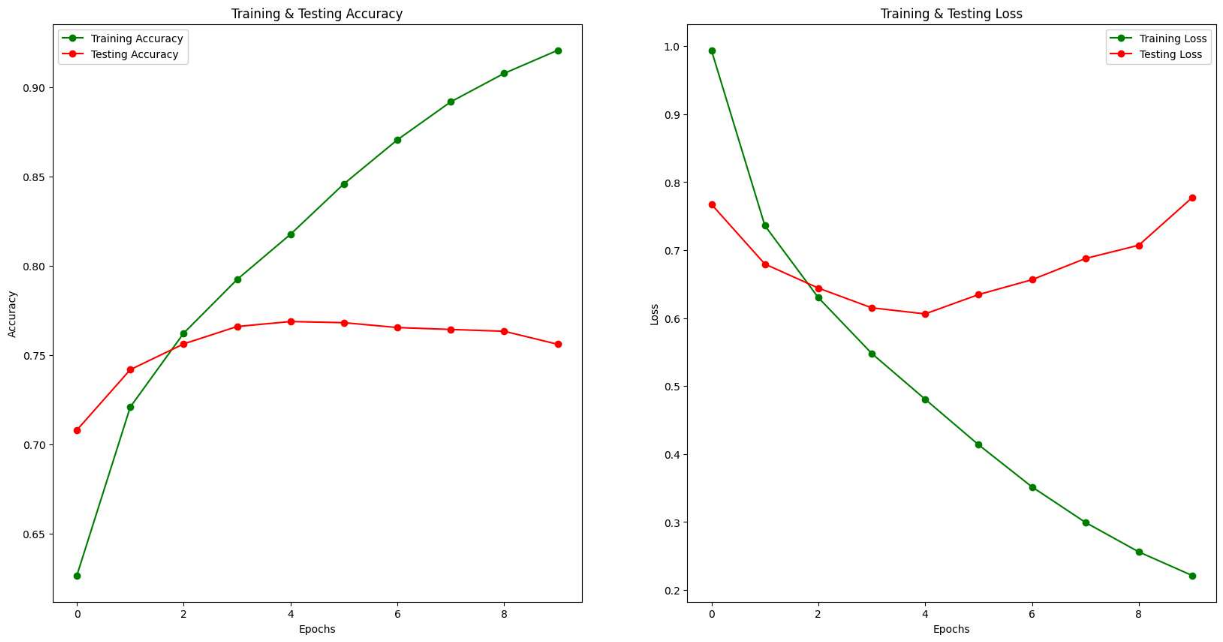
In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, df['status'] , test_size = 0
```

In [19]:
```
history = model.fit(X_train, y_train, batch_size = 128, validation_data = (X_test,y
Epoch 1/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 221s 739ms/step - accuracy: 0.5604 - loss: 1.1916 - val
_accuracy: 0.7081 - val_loss: 0.7674
Epoch 2/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 144s 498ms/step - accuracy: 0.7154 - loss: 0.7514 - val
_accuracy: 0.7420 - val_loss: 0.6795
Epoch 3/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 462s 2s/step - accuracy: 0.7595 - loss: 0.6360 - val_ac
curacy: 0.7564 - val_loss: 0.6442
Epoch 4/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 426s 1s/step - accuracy: 0.7922 - loss: 0.5513 - val_ac
curacy: 0.7661 - val_loss: 0.6151
Epoch 5/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 371s 1s/step - accuracy: 0.8177 - loss: 0.4810 - val_ac
curacy: 0.7689 - val_loss: 0.6062
Epoch 6/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 215s 701ms/step - accuracy: 0.8437 - loss: 0.4208 - val
_accuracy: 0.7683 - val_loss: 0.6346
Epoch 7/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 142s 491ms/step - accuracy: 0.8706 - loss: 0.3529 - val
_accuracy: 0.7656 - val_loss: 0.6565
Epoch 8/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 164s 565ms/step - accuracy: 0.8923 - loss: 0.3000 - val
_accuracy: 0.7645 - val_loss: 0.6877
Epoch 9/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 160s 554ms/step - accuracy: 0.9119 - loss: 0.2477 - val
_accuracy: 0.7635 - val_loss: 0.7072
Epoch 10/10
289/289 ━━━━━━━━━━━━━━━━━━━━ 485s 2s/step - accuracy: 0.9255 - loss: 0.2140 - val_ac
curacy: 0.7562 - val_loss: 0.7773
```

In [20]:
```python
epochs = [i for i in range(10)]
fig, ax = plt.subplots(1, 2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
fig.set_size_inches(20, 10)

ax[0].plot(epochs, train_acc, 'go-', label='Training Accuracy')
ax[0].plot(epochs, val_acc, 'ro-', label='Testing Accuracy')
ax[0].set_title('Training & Testing Accuracy')
ax[0].legend()
ax[0].set_xlabel("Epochs")
ax[0].set_ylabel("Accuracy")

ax[1].plot(epochs, train_loss, 'go-', label='Training Loss')
ax[1].plot(epochs, val_loss, 'ro-', label='Testing Loss')
ax[1].set_title('Training & Testing Loss')
ax[1].legend()
ax[1].set_xlabel("Epochs")
ax[1].set_ylabel("Loss")
plt.show()
```



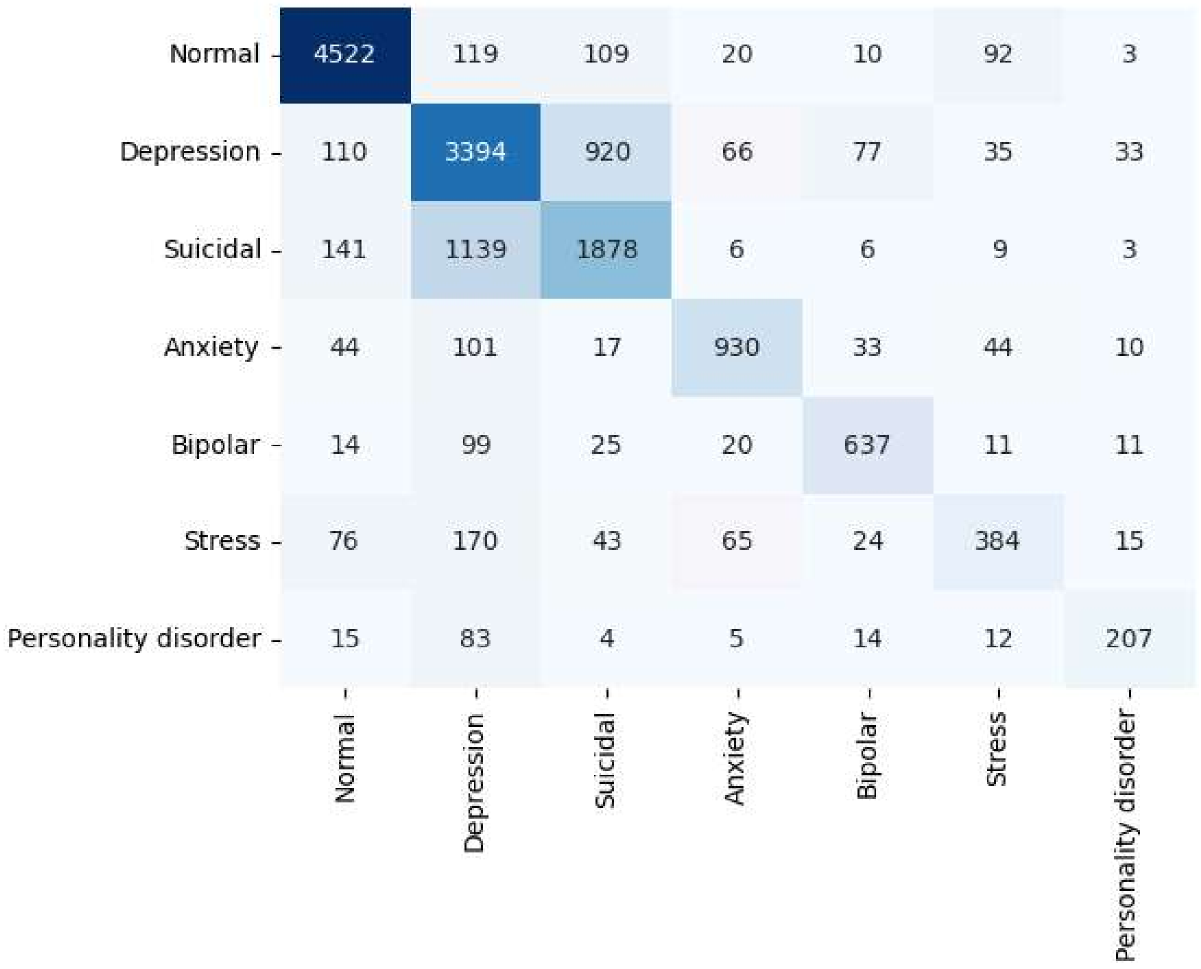
```
In [21]: pred = model.predict(X_test)
         pred_classes = np.argmax(pred, axis=1)

         cm = confusion_matrix(y_test, pred_classes)

         sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', cbar=False,
                     xticklabels=['Normal', 'Depression', 'Suicidal', 'Anxiety', 'Bipolar',
                     yticklabels=['Normal', 'Depression', 'Suicidal', 'Anxiety', 'Bipolar',


         plt.show()
```

```
494/494 ━━━━━━━━━━━━━━━━━━━━ 13s 25ms/step
```



	Normal	Depression	Suicidal	Anxiety	Bipolar	Stress	Personality disorder
Normal	4522	119	109	20	10	92	3
Depression	110	3394	920	66	77	35	33
Suicidal	141	1139	1878	6	6	9	3
Anxiety	44	101	17	930	33	44	10
Bipolar	14	99	25	20	637	11	11
Stress	76	170	43	65	24	384	15
Personality disorder	15	83	4	5	14	12	207