In [99]:

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.renderers.default="iframe"
import seaborn as sns
%matplotlib inline
import string
import re
import random
from imblearn.over_sampling import RandomOverSampler
from scipy.sparse import hstack  # To combine sparse matrices
from wordcloud import WordCloud
from nltk.corpus import stopwords
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import pos_tag
from collections import defaultdict

from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding, Conv1D, GlobalMaxPooling
1D
import keras_tuner as kt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

```
import tensorflow as tf

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from scipy.sparse import hstack
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import LSTM, Dense, Embedding, Dropout, Inpu
t, Concatenate
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
import keras_tuner as kt

import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

import warnings
warnings.filterwarnings("ignore")
```

Loading Data

In [100]:

```
data=pd.read_csv('/kaggle/input/sentiment-analysis-for-mental-health/Comb
ined Data.csv')
```

Preprocessing

In [101]:
```
data['status'].value_counts()
```

Out[101]:
```
status
Normal                  16351
Depression              15404
Suicidal                10653
Anxiety                  3888
Bipolar                  2877
Stress                   2669
Personality disorder     1201
Name: count, dtype: int64
```

In [102]:
```
data.drop("Unnamed: 0",axis=1,inplace=True)
```

Detection and Removing Missing Values

In [103]:
```
data.isnull().sum()
```

Out[103]:
```
statement    362
status         0
dtype: int64
```

In [104]:
```
data=data.dropna()
```

In [105]:
```
data.isnull().sum()
```

Out[105]:
```
statement    0
status       0
dtype: int64
```

In [106]:
```
data
```

Out[106]:

	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
...	...	...
53038	Nobody takes me seriously I've (24M) dealt wit...	Anxiety
53039	selfishness "I don't feel very good, it's lik...	Anxiety
53040	Is there any way to sleep better? I can't slee...	Anxiety
53041	Public speaking tips? Hi, all. I have to give ...	Anxiety
53042	I have really bad door anxiety! It's not about...	Anxiety

52681 rows × 2 columns

In [ ]:

EDA

In [107]:
```
data.describe()
```

Out[107]:

	statement	status
count	52681	52681
unique	51073	7
top	what do you mean?	Normal
freq	22	16343

In [108]:

```
status=data['status'].value_counts().unique()
status
```

Out[108]:

```
array([16343, 15404, 10652,  3841,  2777,  2587,  1077])
```

In [109]:

```
labels=['Anxiety','Normal','Depression','Suicidal','Stress','Bipolar','Pe
rsonality Disorder']
colors=['Gold','mediumturquoise','darkorange','lightgreen','saddlebrow
n','rebeccapurple']

fig = go.Figure(
    data=[
        go.Pie(
            labels=labels,
            values=status,
            textinfo='label+percent',
            textfont=dict(
                size=15,
                color='black'
            ),
            marker=dict(colors=colors, pattern=dict(shape=[".", "x", "+",
"-",'|','+','x']))
        )
    ]
)

fig.show()
```

In [110]:

```
data['num_of_words'] = data['statement'].apply(lambda x: len(str(x).split
()))
data['num_of_sentence']=data['statement'].apply(lambda x: len(nltk.sent_t
okenize(x)))
data['num_of_characters']=data['statement'].apply(len)
data
```

Out[110]:

	statement	status	num_of_words	num_of_sentence	num_of_characters
0	oh my gosh	Anxiety	3	1	10
1	trouble sleeping, confused mind, restless hear...	Anxiety	10	2	64
2	All wrong, back off dear, forward doubt. Stay ...	Anxiety	14	2	78
3	I've shifted my focus to something else but I'...	Anxiety	11	1	61
4	I'm restless and restless, it's been a month n...	Anxiety	14	2	72
...	...	...	...	...	...
53038	Nobody takes me seriously I’ve (24M) dealt wit...	Anxiety	322	16	1766
53039	selfishness "I don't feel very good, it's lik...	Anxiety	198	12	1012
53040	Is there any way to sleep better? I can't slee...	Anxiety	17	2	85
53041	Public speaking tips? Hi, all. I have to give ...	Anxiety	74	6	401
53042	I have really bad door anxiety! It's not about...	Anxiety	79	3	417

52681 rows × 5 columns

In [111]:
```
random_stat=data.groupby('status')['statement'].apply(lambda x: x.sample
(n=1).iloc[0])
for status,statement in random_stat.items():
    print(f'Status: {status}')
    print(f'Statement: {statement}')
```

Status: Anxiety
Statement: finger nails... beaus lines So recently within the past few months i have started to develop horizontal ridges on my finger and toe nails. I googled and discovered they are called beaus lines and they often indicate different diseases. I'm in pretty good health and eat well and exercise often but I'm still concerned as to why i have these horizontal ridges on my nails?? please help if you have any insight
Status: Bipolar
Statement: Realizations In a very bad depression now. And I don't know if it's the depression or what, but I'm am realizing most of my family and friends treat me like shit and really take me for granted. My husband, my kids, hell even my mom. Starting to realize my support system isn't that great. That only makes my depression worse!
Status: Depression
Statement: Every day around my family I am innable to show any emotion. In home it is like "hot and cold" they can yell at me and tell me how useless I am and then because I guess they feel guilty they either apologise or try changing the subject . I have only one friend which I really REALLY care for but their life is much more complicated and I feel like shit ...I cannot help I can"t do amything they want to commits. and I am unable to stop that . I know nothing ,I do nothing , I have nothing... the only thing I feel is confusion ,emptyness or irritation I do not know what is happening to me anymore
Status: Normal
Statement: Watching: KAWAJI ZONE+
Status: Personality disorder
Statement: My first therapy appointment in a long time I have had bad luck with therapists in the past but decided to try again. I am mainly going to overcome a fear of driving. I have driven before back as a teenager and the process was going well but there were some things that happened that then hindered it. Before in therapy, I didn't have specific goals and I don't think that helped.
Status: Stress
Statement: I'm scared I'm going to slip the next time I get in the car or on my motorcycle and just plow into something at full speed just because it'd be so easy and it might end all the pain. But then I don't want to do that, because I have a safe car, and I wear full safety gear when I ride, so those are both maybes. I don't have a gun. I find my thoughts straying from just using one if I had one to thinking of places I could buy one from. Does Walmart sell them in California?

```
Status: Suicidal
Statement: My life has gotten worse for 10 years. Nothing has worked t
o make it better. I just cannot do this anymore, I am tired..I am just
so fucking tired of every day trying to cope with everything. I have n
ot lived in so long. My mental health and chronic pain have completely
engulfed my life. I am a she will of who I once was. I am a burden to
my family and girlfriend. I do not even know if she loves me anymore b
ut I cannot blame her. I have lost myself. I have lost her. I have nob
ody. I cannot FUCKING DO THIS
```

In [112]:

```
avg_word=data.groupby('status')['num_of_words'].mean().reset_index()
avg_word
```

Out[112]:

	status	num_of_words
0	Anxiety	143.837022
1	Bipolar	176.200216
2	Depression	168.021488
3	Normal	17.246283
4	Personality disorder	179.346332
5	Stress	114.585234
6	Suicidal	146.440293

In [113]:

```
avg_sent=data.groupby('status')['num_of_sentence'].mean().reset_index()
avg_sent
```

Out[113]:

	status	num_of_sentence
0	Anxiety	8.490758
1	Bipolar	11.046093
2	Depression	7.830369
3	Normal	1.522793
4	Personality disorder	10.675023
5	Stress	6.474681
6	Suicidal	8.794217

In [114]:

```
avg_char=data.groupby('status')['num_of_characters'].mean().reset_index()
avg_char
```

Out[114]:

	status	num_of_characters
0	Anxiety	764.681073
1	Bipolar	946.074901
2	Depression	844.031161
3	Normal	90.244447
4	Personality disorder	956.731662
5	Stress	613.545419
6	Suicidal	734.967330

In [115]:

```
col=['num_of_words','num_of_sentence','num_of_characters']
```

In [116]:

```
melted_data=data[col].melt(var_name='Variable', value_name='Value')
```

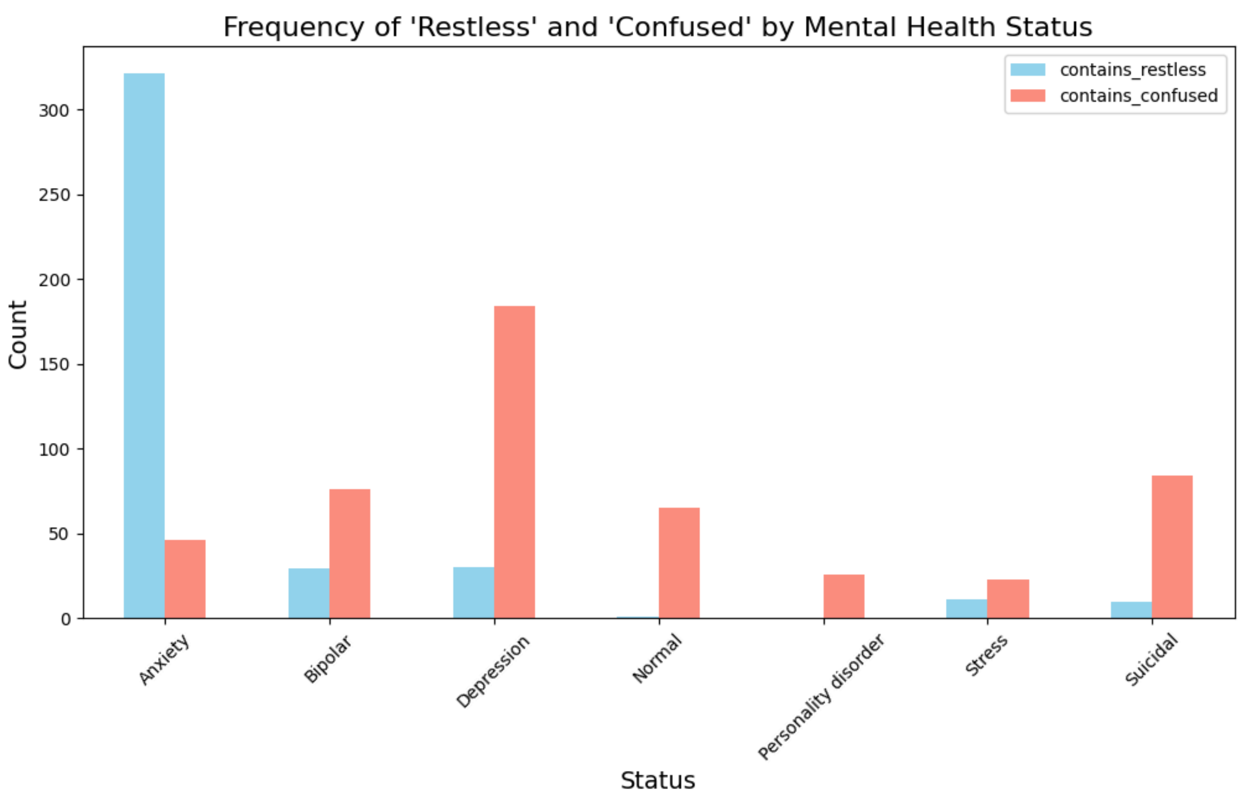
In [117]:
```
fig= px.box(melted_data,y='Variable',x='Value', title='Distribution of Nu
m of Statements',color='Variable')
fig.update_layout(xaxis_title='Variables',yaxis_title='Values', showlegen
d=True)
fig.layout.legend.x=1
fig.layout.legend.y=1
fig.layout.legend.xref='paper'
fig.layout.legend.yref='paper'
fig.layout.legend.font.size=12
fig.layout.legend.title = 'Column Names'
fig.layout.legend.bgcolor='aliceblue'
fig.layout.legend.bordercolor = 'indigo'
fig.layout.legend.borderwidth = 4

fig.show()
```

In [118]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=avg_word['status'],
    y=avg_word['num_of_words'],
    mode='lines+markers',
    name='Words',
))

fig.add_trace(go.Scatter(
    x=avg_sent['status'],
    y=avg_sent['num_of_sentence'],
    mode='lines+markers',
    name='Sentence',
))

fig.add_trace(go.Scatter(
    x=avg_char['status'],
    y=avg_char['num_of_characters'],
    mode='lines+markers',
    name='Characters',
))
fig.update_layout(
    title='Mean of Number of Words, Sentences, and Characters',
    xaxis_title='Status',
    yaxis_title='Mean Count',
    xaxis=dict(tickmode='linear', tick0=0, dtick=1),
    legend=dict(title='Column Name'),
    hovermode='x'
)

fig.show()
```

In [119]:

```
data['contains_restless'] = data['statement'].dropna().apply(lambda x: 'restless' in x.lower())
data['contains_confused'] = data['statement'].dropna().apply(lambda x: 'confused' in x.lower())

word_counts = data.groupby('status')[['contains_restless', 'contains_confused']].sum()
word_counts.plot(kind='bar', figsize=(12, 6), color=['skyblue', 'salmon'], alpha=0.9)
plt.title("Frequency of 'Restless' and 'Confused' by Mental Health Status", fontsize=16)
plt.xlabel("Status", fontsize=14)
plt.ylabel("Count", fontsize=14)
plt.xticks(rotation=45)
plt.show()
```



Lower Casing

In [120]:

```
data['statement']=data['statement'].str.lower()
data
```

Out[120]:

	statement	status	num_of_words	num_of_sentence	num_of_characters	contains_restless
0	oh my gosh	Anxiety	3	1	10	False
1	trouble sleeping, confused mind, restless hear...	Anxiety	10	2	64	True
2	all wrong, back off dear, forward doubt. stay ...	Anxiety	14	2	78	True
3	i've shifted my focus to something else but i'...	Anxiety	11	1	61	False
4	i'm restless and restless, it's been a month n...	Anxiety	14	2	72	True
...	...	...	...	...	...	...
53038	nobody takes me seriously i’ve (24m) dealt wit...	Anxiety	322	16	1766	False
53039	selfishness "i don't feel very good, it's lik...	Anxiety	198	12	1012	False
53040	is there any way to sleep better? i can't slee...	Anxiety	17	2	85	False
53041	public speaking tips? hi, all. i have to give ...	Anxiety	74	6	401	False
53042	i have really bad door anxiety! it's not about...	Anxiety	79	3	417	False

52681 rows × 7 columns

Removing Punctuation, Unicode Characters, Unwanted Patterns and URL

In [121]:

```
def removing(text):
    #for removing URLs
    text=re.sub(r'http[s]?://\S+', '', text)
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    #removing markdown-style links
    text=re.sub(r'\[.*?\]\(.*?\)','',text)
    #removing square brackets
    text = re.sub(r'\[.*?\]', '', text)
    #remove HTML tags
    text = re.sub(r'<.*?>+', '', text)
    #remove@
    text=re.sub(r'@\w+','',text)
    #remove punctuation and special characters
    text=re.sub(r'[^\w\s]','',text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)
    #removing age and gender patterns like 24M or 30F
    text=re.sub(r'\b\d{1,3}[mf]\b','',text)
    text=re.sub(r'\b\d{1,3}[MF]\b','',text)

    pattern = r'\b(?:ive|im|hes|shes|theyre|weve|youve|cant|wont|isnt|are
nt|wasnt|hi|werent|havent|hasnt|didnt|doesnt|youre|id|hed|dont|else|shed|
theyd|itd|wed|youd)\b'

    # Remove the matching contractions (case-insensitive)
    text = re.sub(pattern, '', text, flags=re.IGNORECASE)



    #removing new lines
    text = re.sub(r'\n', '', text)
    return text.strip()

data['statement']=data['statement'].apply(removing)
data.statement
```

Out[121]:

```
0                                                  oh my gosh
1        trouble sleeping confused mind restless heart ...
2        all wrong back off dear forward doubt stay in ...
3        shifted my focus to something  but  still worried
4        restless and restless its been a month now boy...
                                ...
53038    nobody takes me seriously   dealt with depress...
53039    selfishness  i  feel very good its like i  bel...
53040    is there any way to sleep better i  sleep most...
53041    public speaking tips  all i have to give a pre...
53042    i have really bad door anxiety its not about b...
Name: statement, Length: 52681, dtype: object
```

In [122]:
```
stop_words=set(stopwords.words('english'))
def remove_stopwords(text):
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)

data['cleaned_statement'] = data['statement'].apply(lambda x: remove_stop
words(x))
data
```

Out[122]:

	statement	status	num_of_words	num_of_sentence	num_of_characters	contains_restless
0	oh my gosh	Anxiety	3	1	10	False
1	trouble sleeping confused mind restless heart ...	Anxiety	10	2	64	True
2	all wrong back off dear forward doubt stay in ...	Anxiety	14	2	78	True
3	shifted my focus to something but still worried	Anxiety	11	1	61	False
4	restless and restless its been a month now boy...	Anxiety	14	2	72	True
...	...	...	...	...	...	...
53038	nobody takes me seriously dealt with depress...	Anxiety	322	16	1766	False
53039	selfishness i feel very good its like i bel...	Anxiety	198	12	1012	False
53040	is there any way to sleep better i sleep most...	Anxiety	17	2	85	False
53041	public speaking tips all i have to give a pre...	Anxiety	74	6	401	False
53042	i have really bad door anxiety its not about b...	Anxiety	79	3	417	False

52681 rows × 8 columns

In [123]:
```
data['tokens']=data['cleaned_statement'].apply(word_tokenize)
data
```

Out[123]:

	statement	status	num_of_words	num_of_sentence	num_of_characters	contains_restless
0	oh my gosh	Anxiety	3	1	10	False
1	trouble sleeping confused mind restless heart ...	Anxiety	10	2	64	True
2	all wrong back off dear forward doubt stay in ...	Anxiety	14	2	78	True
3	shifted my focus to something but still worried	Anxiety	11	1	61	False
4	restless and restless its been a month now boy...	Anxiety	14	2	72	True
...	...	...	...	...	...	...
53038	nobody takes me seriously dealt with depress...	Anxiety	322	16	1766	False
53039	selfishness i feel very good its like i bel...	Anxiety	198	12	1012	False
53040	is there any way to sleep better i sleep most...	Anxiety	17	2	85	False
53041	public speaking tips all i have to give a pre...	Anxiety	74	6	401	False
53042	i have really bad door anxiety its not about b...	Anxiety	79	3	417	False

52681 rows × 9 columns

POS Analysis

In [124]:

```
def pos_analysis(tokens):

    pos_mapping = {
        'JJ': 'Adjective', 'JJR': 'Adjective', 'JJS': 'Adjective',
        'NN': 'Noun', 'NNS': 'Noun', 'NNP': 'Noun', 'NNPS': 'Noun',
        'VB': 'Verb', 'VBD': 'Verb', 'VBG': 'Verb', 'VBN': 'Verb', 'VBP':
'Verb', 'VBZ': 'Verb',
        'RB': 'Adverb', 'RBR': 'Adverb', 'RBS': 'Adverb',
    }

    tagged = pos_tag(tokens)

    pos_count = defaultdict(int)
    pos_words = defaultdict(list)

    for word, tag in tagged:

        if tag in pos_mapping:
            category = pos_mapping[tag]
            pos_count[category] += 1
            pos_words[category].append(word)

    return {
        'pos_count': pos_count,
        'pos_words': pos_words,
    }
```

In [125]:
```
def analyze_row(row):
    result = pos_analysis(row)

    return pd.Series({
        'Adjectives': ', '.join(result['pos_words'].get('Adjective',
[])),
        'Adjective_Count': result['pos_count'].get('Adjective', 0),
        'Nouns': ', '.join(result['pos_words'].get('Noun', [])),
        'Noun_Count': result['pos_count'].get('Noun', 0),
        'Verbs': ', '.join(result['pos_words'].get('Verb', [])),
        'Verb_Count': result['pos_count'].get('Verb', 0),
        'Adverbs': ', '.join(result['pos_words'].get('Adverb', [])),
        'Adverb_Count': result['pos_count'].get('Adverb', 0),
    })

pos_results = data['tokens'].apply(analyze_row)

data = pd.concat([data, pos_results], axis=1)
```

In [126]:

```
pd.set_option('display.max_columns', None)
data.head()
```

Out[126]:

	statement	status	num_of_words	num_of_sentence	num_of_characters	contains_restless	conta
0	oh my gosh	Anxiety	3	1	10	False	False
1	trouble sleeping confused mind restless heart ...	Anxiety	10	2	64	True	True
2	all wrong back off dear forward doubt stay in ...	Anxiety	14	2	78	True	False
3	shifted my focus to something but still worried	Anxiety	11	1	61	False	False
4	restless and restless its been a month now boy...	Anxiety	14	2	72	True	False

In [127]:

```
numeric_cols = data.select_dtypes(include=['int', 'float']).columns
for col in numeric_cols:
    print(col, (data[col] == 0).sum())
```

```
num_of_words 0
num_of_sentence 0
num_of_characters 0
Adjective_Count 7900
Noun_Count 1053
Verb_Count 6146
Adverb_Count 14463
```

In [128]:
```
pos_cols = [
    'Adjective_Count',
    'Noun_Count',
    'Verb_Count',
    'Adverb_Count'
]
```

In [129]:
```
data_avg = data.groupby('status')[pos_cols + ['num_of_words']].mean().reset_index()
```

In [130]:
```
for col in pos_cols:
    ratio_col = col + '_ratio'
    data_avg[ratio_col] = data_avg[col] / data_avg['num_of_words']
```

In [131]:
```
data_avg
```

Out[131]:

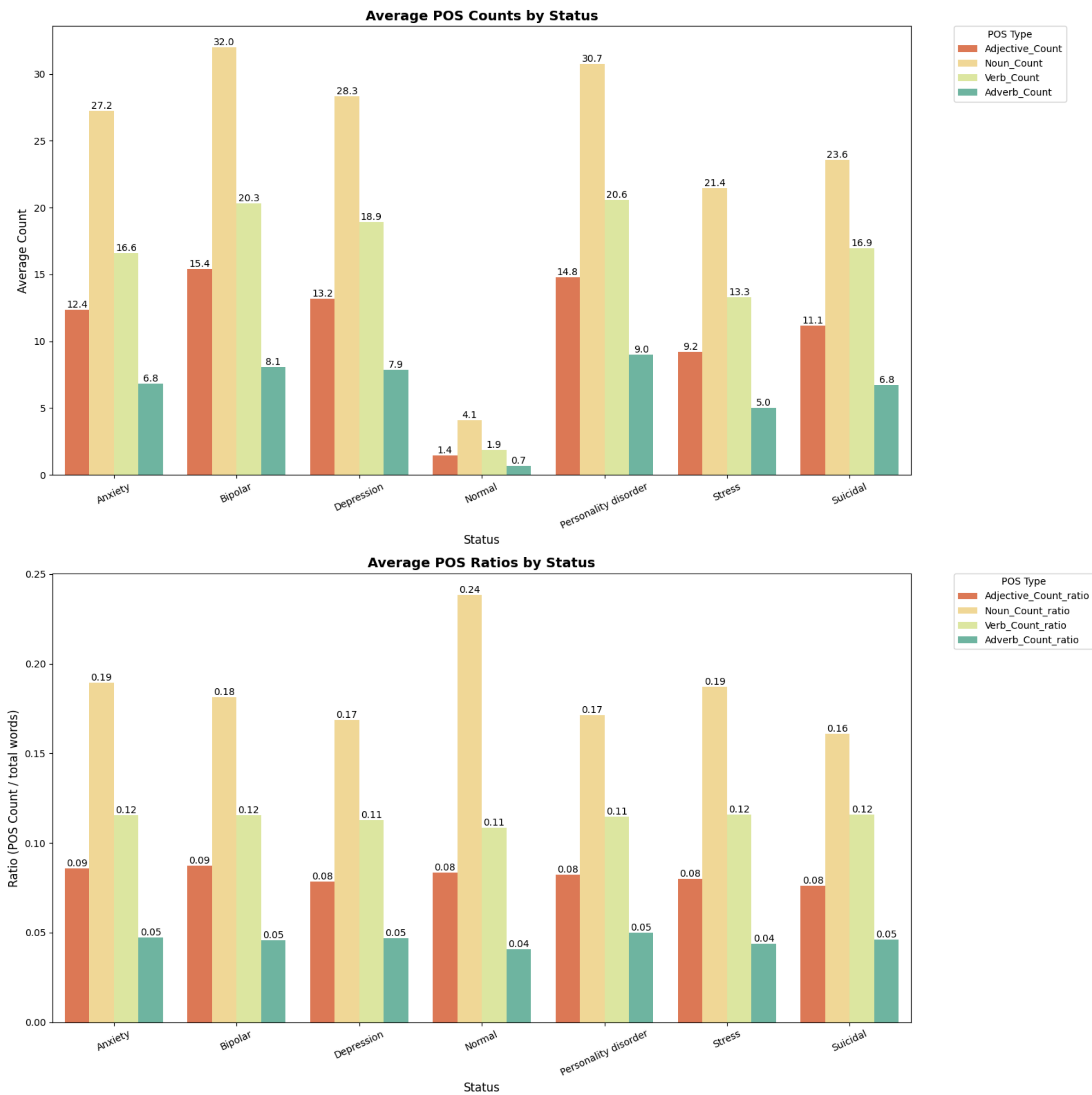
	status	Adjective_Count	Noun_Count	Verb_Count	Adverb_Count	num_of_words	Adjective_(
0	Anxiety	12.360844	27.221297	16.610518	6.813330	143.837022	0.085936
1	Bipolar	15.395031	31.979834	20.339215	8.074181	176.200216	0.087372
2	Depression	13.160997	28.320696	18.935666	7.857310	168.021488	0.078329
3	Normal	1.441106	4.111913	1.871015	0.700973	17.246283	0.083560
4	Personality disorder	14.773445	30.737233	20.568245	8.984215	179.346332	0.082374
5	Stress	9.185543	21.439505	13.271357	5.020487	114.585234	0.080163
6	Suicidal	11.149268	23.557360	16.938415	6.752816	146.440293	0.076135

In [132]:
```
pos_ratio_cols = [col + '_ratio' for col in pos_cols]

melted_counts = data_avg.melt(
    id_vars='status',
    value_vars=pos_cols,
    var_name='pos_type',
    value_name='average_count'
)

melted_ratios = data_avg.melt(
    id_vars='status',
    value_vars=pos_ratio_cols,
    var_name='pos_type',
    value_name='ratio'
)
```

In [133]:

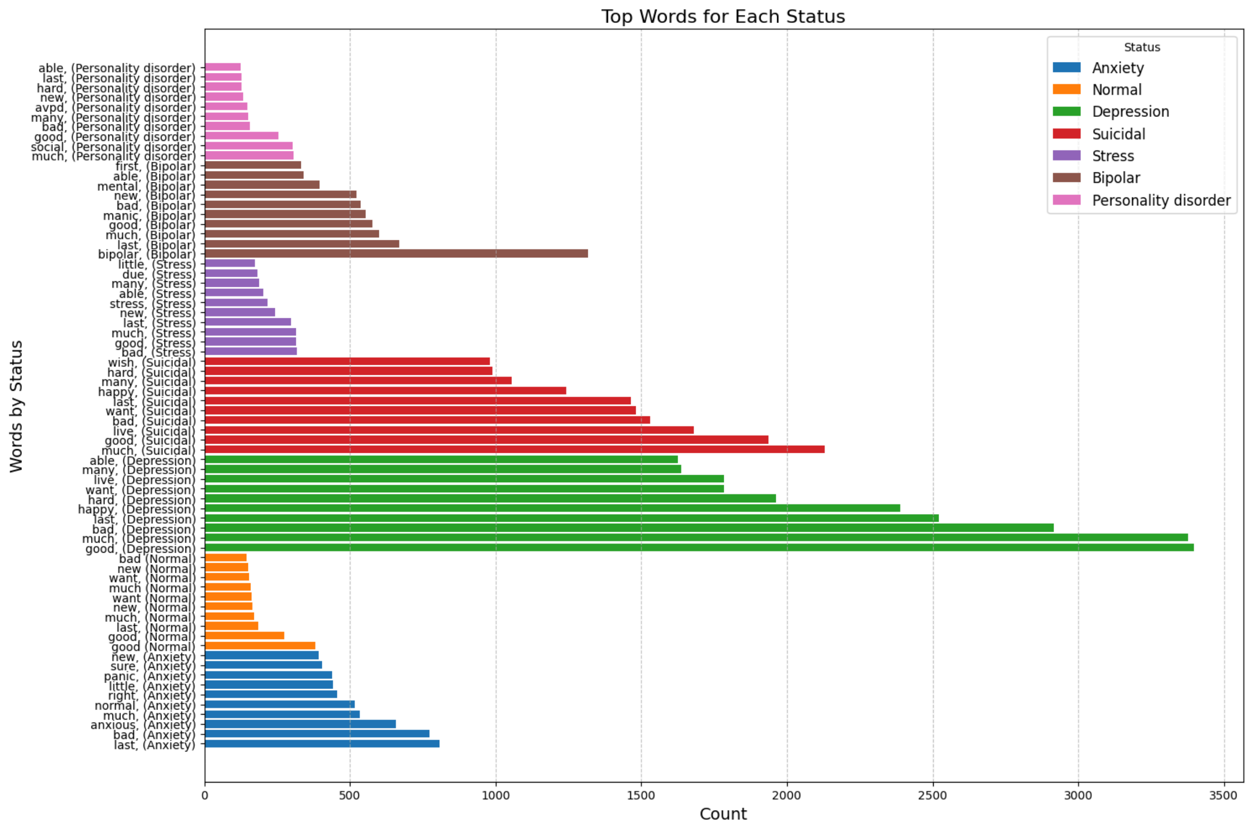
```
fig, axes = plt.subplots(2, 1, figsize=(16, 16))

palette = sns.color_palette("Spectral", n_colors=len(pos_cols))

sns.barplot(
    data=melted_counts,
    x='status',
    y='average_count',
    hue='pos_type',
    ax=axes[0],
    palette=palette
)
axes[0].set_title('Average POS Counts by Status', fontsize=14, fontweight
='bold')
axes[0].set_xlabel('Status', fontsize=12)
axes[0].set_ylabel('Average Count', fontsize=12)
axes[0].tick_params(axis='x', rotation=25)

for container in axes[0].containers:
    axes[0].bar_label(container, fmt="%.1f", label_type="edge", fontsize=
10)

axes[0].legend(title='POS Type', bbox_to_anchor=(1.05, 1), loc='upper lef
t', borderaxespad=0)

sns.barplot(
    data=melted_ratios,
    x='status',
    y='ratio',
    hue='pos_type',
    ax=axes[1],
    palette=palette
)
axes[1].set_title('Average POS Ratios by Status', fontsize=14, fontweight
='bold')
axes[1].set_xlabel('Status', fontsize=12)
axes[1].set_ylabel('Ratio (POS Count / total words)', fontsize=12)
axes[1].tick_params(axis='x', rotation=25)

for container in axes[1].containers:
```

```
    axes[1].bar_label(container, fmt="%.2f", label_type="edge", fontsize=10)

axes[1].legend(title='POS Type', bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)

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



In [134]:

```
plot_data = []


statuses = data['status'].dropna().unique()


for status in statuses:
    words = ' '.join(data[data['status'] == status]['Adjectives'].dropna
()).split()
    word_counts = Counter(words).most_common(10)
    for word, count in word_counts:
        plot_data.append({'status': status, 'word': word, 'count': coun
t})


word_df = pd.DataFrame(plot_data)


plt.figure(figsize=(15, 10))
statuses = word_df['status'].unique()

for i, status in enumerate(statuses):
    subset = word_df[word_df['status'] == status]
    plt.barh(subset['word'] + f" ({status})", subset['count'], label=stat
us)

plt.title("Top Words for Each Status", fontsize=16)
plt.xlabel("Count", fontsize=14)
plt.ylabel("Words by Status", fontsize=14)
plt.legend(title="Status", loc='upper right', fontsize=12)
plt.tight_layout()
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```


Top Words for Each Status
Status
Anxiety
Normal
Depression
Suicidal
Stress
Bipolar
Personality disorder
Words by Status
Count
0
500
1000
1500
2000
2500
3000
3500

In [ ]:
```
import math

num_cols = 2
num_rows = math.ceil(len(statuses) / num_cols)
plt.figure(figsize=(14, 4 * num_rows))

for i, status in enumerate(statuses):

    df_sub = data.loc[data['status'] == status, 'cleaned_statement'].drop
na()

    text_all = " ".join(df_sub.tolist())

    vectorizer = CountVectorizer(ngram_range=(2,2), max_features=50)
    X = vectorizer.fit_transform([text_all])
    bigram_freqs = X.toarray()[0]
    bigram_features = vectorizer.get_feature_names_out()

    freq_dict = dict(zip(bigram_features, bigram_freqs))

    if sum(freq_dict.values()) == 0:
        print(f"No bigrams found for status '{status}'. Skipping.")
        continue

    wordcloud = WordCloud(
        width=800,
        height=400,
        background_color='white',
        colormap='plasma'
    ).generate_from_frequencies(freq_dict)

    ax = plt.subplot(num_rows, num_cols, i + 1)
    ax.imshow(wordcloud, interpolation='bilinear')
    ax.set_title(f"Bigrams WordCloud: {status}", fontsize=14, pad=10)
    ax.axis("off")

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

In [136]:

```
data
```

Out[136]:

	statement	status	num_of_words	num_of_sentence	num_of_characters	contains_restless
0	oh my gosh	Anxiety	3	1	10	False
1	trouble sleeping confused mind restless heart ...	Anxiety	10	2	64	True
2	all wrong back off dear forward doubt stay in ...	Anxiety	14	2	78	True
3	shifted my focus to something but still worried	Anxiety	11	1	61	False
4	restless and restless its been a month now boy...	Anxiety	14	2	72	True
...	...	...	...	...	...	...
53038	nobody takes me seriously dealt with depress...	Anxiety	322	16	1766	False
53039	selfishness i feel very good its like i bel...	Anxiety	198	12	1012	False
53040	is there any way to sleep better i sleep most...	Anxiety	17	2	85	False
53041	public speaking tips all i have to give a pre...	Anxiety	74	6	401	False

	statement	status	num_of_words	num_of_sentence	num_of_characters	contains_restless
53042	i have really bad door anxiety its not about b...	Anxiety	79	3	417	False

52681 rows × 17 columns

In [137]:

```
le = LabelEncoder()
data['status_label'] = le.fit_transform(data['status'])
```

In [138]:

```
data.status_label.unique()
```

Out[138]:

```
array([0, 3, 2, 6, 5, 1, 4])
```

In [139]:

```
data.status.unique()
```

Out[139]:

```
array(['Anxiety', 'Normal', 'Depression', 'Suicidal', 'Stress', 'Bipol
ar',
       'Personality disorder'], dtype=object)
```

After Label Encoding to Status

- Anxiety -> 0
- Bipolar -> 1
- Depression -> 2
- Normal -> 3
- Personal Disorder -> 4
- Stress -> 5
- Suicidal -> 6

In [140]:
```
data.columns
```

Out[140]:
```
Index(['statement', 'status', 'num_of_words', 'num_of_sentence',
       'num_of_characters', 'contains_restless', 'contains_confused',
       'cleaned_statement', 'tokens', 'Adjectives', 'Adjective_Count',
'Nouns',
       'Noun_Count', 'Verbs', 'Verb_Count', 'Adverbs', 'Adverb_Count',
       'status_label'],
      dtype='object')
```

We need to make label encoding Adjectives, Nouns, Verbs, Adverbs colums to find which of them used for detecting to status.

In [141]:
```
data['Adj_le'] = le.fit_transform(data['Adjectives'])
data['N_le'] = le.fit_transform(data['Nouns'])
data['V_le'] = le.fit_transform(data['Verbs'])
data['Adv_le'] = le.fit_transform(data['Adverbs'])
```

In [142]:
```
data_copy=data[[ 'status', 'num_of_words', 'num_of_sentence',
       'num_of_characters',
       'cleaned_statement', 'Adjective_Count',
       'Noun_Count', 'Verb_Count', 'Adverb_Count', 'Adj_le', 'N_le', 'V_l
e', 'Adv_le']]
```

In [143]:

```
data_copy
```

Out[143]:

	status	num_of_words	num_of_sentence	num_of_characters	cleaned_statement	Adjective_(
0	Anxiety	3	1	10	oh gosh	0
1	Anxiety	10	2	64	trouble sleeping confused mind restless heart ...	1
2	Anxiety	14	2	78	wrong back dear forward doubt stay restless re...	3
3	Anxiety	11	1	61	shifted focus something still worried	0
4	Anxiety	14	2	72	restless restless month boy mean	2
...	...	...	...	...	...	...
53038	Anxiety	322	16	1766	nobody takes seriously dealt depressionanxiety...	30
53039	Anxiety	198	12	1012	selfishness feel good like belong world think ...	19
53040	Anxiety	17	2	85	way sleep better sleep nights meds help	2
53041	Anxiety	74	6	401	public speaking tips give presentation work ne...	8
53042	Anxiety	79	3	417	really bad door anxiety scared lock door somet...	9

52681 rows × 13 columns

In [144]:

```
feature_cols = [
    'Adjective_Count','Noun_Count','Verb_Count', 'Adverb_Count',
    'num_of_words', 'num_of_sentence', 'num_of_characters'
]

X_fe = data[feature_cols]
y_fe = data['status_label']
```

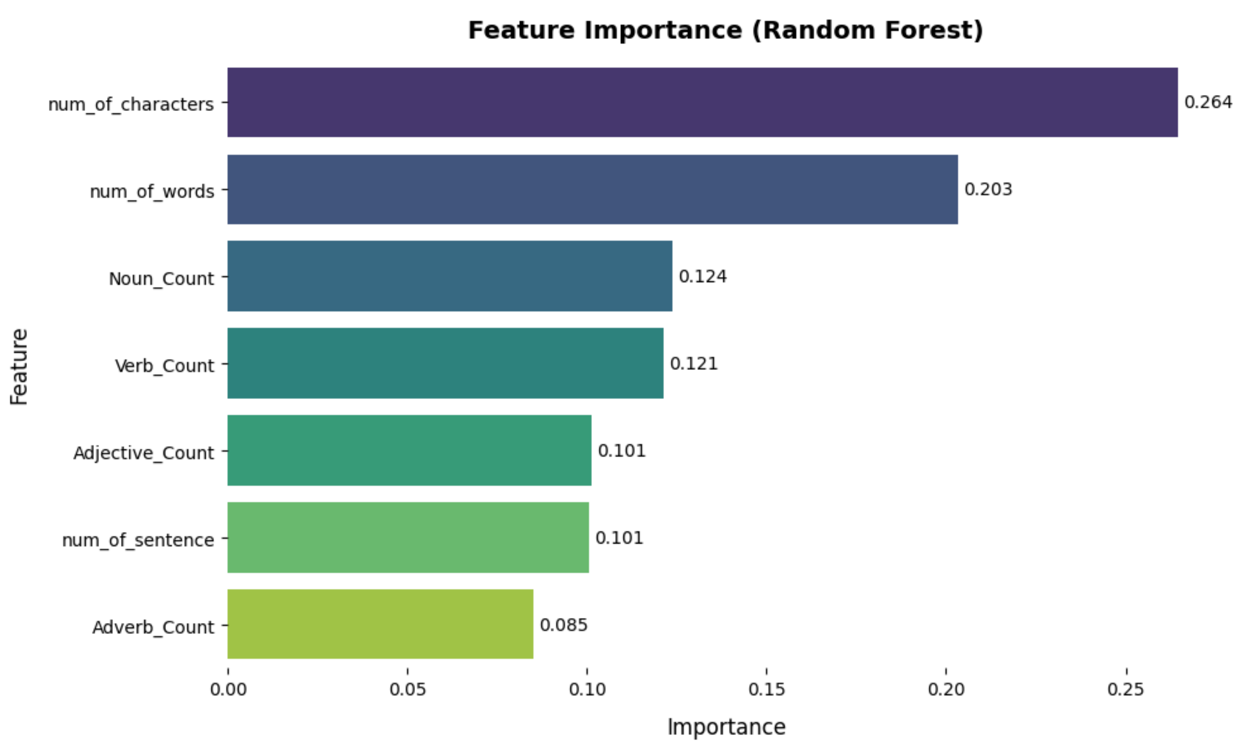
In [145]:

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_fe, y_fe)

importances = rf.feature_importances_

data_importances = pd.DataFrame({
    'feature': feature_cols,
    'importance': importances
}).sort_values('importance', ascending=False)

data_importances
```

Out[145]:

	feature	importance
6	num_of_characters	0.264471
4	num_of_words	0.203271
1	Noun_Count	0.123722
2	Verb_Count	0.121228
0	Adjective_Count	0.101448
5	num_of_sentence	0.100645
3	Adverb_Count	0.085215

In [146]:
```
plt.figure(figsize=(10, 6))

g = sns.barplot(
    data=data_importances,
    x='importance',
    y='feature',
    palette='viridis'
)

plt.title('Feature Importance (Random Forest)', fontsize=14, fontweight
='bold', pad=12)
plt.xlabel('Importance', fontsize=12, labelpad=10)
plt.ylabel('Feature', fontsize=12, labelpad=10)

for container in g.containers:
    g.bar_label(container, fmt='%.3f', label_type='edge', padding=3)

sns.despine(left=True, bottom=True)

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



Feature Selection & Spilitting Data as Test and Train

In [147]:
```
df=data[['status','cleaned_statement','num_of_characters','num_of_word
s']]
```

In [148]:
```
df.isnull().sum()
```

Out[148]:
```
status               0
cleaned_statement    0
num_of_characters    0
num_of_words         0
dtype: int64
```

In [149]:
```
df
```

Out[149]:

	status	cleaned_statement	num_of_characters	num_of_words
0	Anxiety	oh gosh	10	3
1	Anxiety	trouble sleeping confused mind restless heart ...	64	10
2	Anxiety	wrong back dear forward doubt stay restless re...	78	14
3	Anxiety	shifted focus something still worried	61	11
4	Anxiety	restless restless month boy mean	72	14
...	...	...	...	...
53038	Anxiety	nobody takes seriously dealt depressionanxiety...	1766	322
53039	Anxiety	selfishness feel good like belong world think ...	1012	198
53040	Anxiety	way sleep better sleep nights meds help	85	17
53041	Anxiety	public speaking tips give presentation work ne...	401	74
53042	Anxiety	really bad door anxiety scared lock door somet...	417	79

52681 rows × 4 columns

In [150]:
```
X_text=df['cleaned_statement']

y=df['status']
```

In [151]:
```
tfidf = TfidfVectorizer(max_features=5000)
X_text_tfidf = tfidf.fit_transform(X_text)
```

In [152]:
```
X_train, X_test, y_train, y_test = train_test_split(X_text_tfidf, y, test
_size=0.2, random_state=42, stratify=y)
```

Model Selection

Logistic Regression Naive Bayes SVM Random Forest XGBoost LightGBM CatBoost LSTM BERT GPT CNN + Global Pooling Pretrained Embedding + CNN

Perform hyperparameter tuning with tools like KerasTuner or GridSearchCV.

Logistic Regression

In [153]:
```
le = LabelEncoder()
```

In [154]:
```
y_train_le=le.fit_transform(y_train)
y_test_le=le.fit_transform(y_test)
```

In [155]:
```
y_train_le
```

Out[155]:
```
array([2, 3, 2, ..., 1, 2, 3])
```

In [156]:
```
lr=LogisticRegression(random_state=42, max_iter=5000)
lr_params={'penalty': ['l1', 'l2','elasticnet'],'C':[0.01, 0.1, 1, 10, 10
0]}
lr_grid=GridSearchCV(lr,lr_params,cv=3,scoring='accuracy')
lr_grid.fit(X_train,y_train_le)
print("Best Logistic Regression Params:", lr_grid.best_params_)
```

```
Best Logistic Regression Params: {'C': 1, 'penalty': 'l2'}
```

In [157]:
```
best_lr_model=lr_grid.best_estimator_
```

In [158]:
```
best_lr_model
```

Out[158]:
```
LogisticRegression
LogisticRegression(C=1, max_iter=5000, random_state=42)
```

In [159]:
```
y_pred_lr=best_lr_model.predict(X_test)
```

In [160]:
```
accuracy_lr=accuracy_score(y_test_le,y_pred_lr)
```

In [161]:
```
print(f"Accuracy of the best Logistic Regression model: {accuracy_lr * 10
0:.2f}%")
```

```
Accuracy of the best Logistic Regression model: 74.83%
```

In [162]:
```
print("Classification Report for Logistic Regression:")
print(classification_report(y_test_le, y_pred_lr))
```

```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.82      0.71      0.76       768
           1       0.89      0.62      0.73       556
           2       0.68      0.74      0.71      3081
           3       0.83      0.95      0.88      3269
           4       0.89      0.40      0.55       215
           5       0.68      0.37      0.47       517
           6       0.67      0.62      0.64      2131

    accuracy                           0.75     10537
   macro avg       0.78      0.63      0.68     10537
weighted avg       0.75      0.75      0.74     10537
```

FNN

X_text=df['cleaned_statement']

y=df['status']

X_train, X_test, y_train, y_test

y_train_le=le.fit_transform(y_train) y_test_le=le.fit_transform(y_test)

In [163]:

```
from tensorflow.keras.optimizers import Adam

nn_model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(len(le.classes_), activation='softmax')
])

optimizer = Adam(learning_rate=0.0005)

# Compile the model with the optimizer
nn_model.compile(optimizer=optimizer,
                 loss='sparse_categorical_crossentropy',
                 metrics=['accuracy'])

nn_history = nn_model.fit(
    X_train.toarray(),
    y_train_le,
    epochs=10,
    batch_size=64,
    validation_split=0.2,
    verbose=1
)


test_loss, test_accuracy = nn_model.evaluate(X_test.toarray(), y_test_le,
verbose=0)


print(f"Test Accuracy: {test_accuracy:.4f}")


from sklearn.metrics import classification_report
y_pred_fnn = nn_model.predict(X_test.toarray(), verbose=0).argmax(axis=1)
print(classification_report(y_test_le, y_pred_fnn, target_names=le.classe
s_))
```

```
Epoch 1/10
527/527 [==============================] - 5s 9ms/step - loss: 1.1614
- accuracy: 0.5753 - val_loss: 0.7784 - val_accuracy: 0.7192
Epoch 2/10
527/527 [==============================] - 4s 8ms/step - loss: 0.7122
- accuracy: 0.7435 - val_loss: 0.6849 - val_accuracy: 0.7494
Epoch 3/10
527/527 [==============================] - 4s 7ms/step - loss: 0.5791
- accuracy: 0.7941 - val_loss: 0.6642 - val_accuracy: 0.7561
Epoch 4/10
527/527 [==============================] - 4s 8ms/step - loss: 0.4980
- accuracy: 0.8223 - val_loss: 0.6694 - val_accuracy: 0.7580
Epoch 5/10
527/527 [==============================] - 4s 8ms/step - loss: 0.4299
- accuracy: 0.8475 - val_loss: 0.6952 - val_accuracy: 0.7529
Epoch 6/10
527/527 [==============================] - 4s 8ms/step - loss: 0.3773
- accuracy: 0.8682 - val_loss: 0.7228 - val_accuracy: 0.7498
Epoch 7/10
527/527 [==============================] - 4s 8ms/step - loss: 0.3292
- accuracy: 0.8852 - val_loss: 0.7522 - val_accuracy: 0.7480
Epoch 8/10
527/527 [==============================] - 4s 8ms/step - loss: 0.2855
- accuracy: 0.9029 - val_loss: 0.8014 - val_accuracy: 0.7461
Epoch 9/10
527/527 [==============================] - 4s 8ms/step - loss: 0.2438
- accuracy: 0.9204 - val_loss: 0.8464 - val_accuracy: 0.7461
Epoch 10/10
527/527 [==============================] - 4s 8ms/step - loss: 0.2153
- accuracy: 0.9298 - val_loss: 0.8812 - val_accuracy: 0.7447
Test Accuracy: 0.7368
                      precision    recall  f1-score   support

             Anxiety       0.79      0.74      0.76       768
             Bipolar       0.81      0.70      0.75       556
          Depression       0.66      0.70      0.68      3081
              Normal       0.88      0.91      0.89      3269
Personality disorder       0.76      0.54      0.63       215
              Stress       0.58      0.49      0.53       517
            Suicidal       0.62      0.62      0.62      2131
```

```
    accuracy                           0.74     10537
   macro avg       0.73      0.67      0.70     10537
weighted avg       0.74      0.74      0.74     10537
```

```
In [ ]:
```

MLP

)

In [164]:

```
from sklearn.neural_network import MLPClassifier

mlp_model = MLPClassifier(hidden_layer_sizes=(128, 64), max_iter=500, ran
dom_state=42)
mlp_model.fit(X_train, y_train_le)

y_pred_mlp = mlp_model.predict(X_test)
print("Neural Network Results:")
print(classification_report( y_test_le, y_pred_mlp))
print("Accuracy:", accuracy_score( y_test_le, y_pred_mlp))
```

```
Neural Network Results:
              precision    recall  f1-score   support

           0       0.80      0.73      0.76       768
           1       0.76      0.72      0.74       556
           2       0.65      0.72      0.68      3081
           3       0.87      0.91      0.89      3269
           4       0.64      0.62      0.63       215
           5       0.57      0.49      0.52       517
           6       0.64      0.55      0.59      2131

    accuracy                           0.73     10537
   macro avg       0.70      0.68      0.69     10537
weighted avg       0.73      0.73      0.73     10537

Accuracy: 0.7310429913637658
```

In [ ]:

LSTM

In [97]:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

max_words=5000
max_sequence_length=50
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_text)
X_text_sequences = tokenizer.texts_to_sequences(X_text)
X_text_padded = pad_sequences(X_text_sequences, maxlen=max_sequence_length,padding='post')

X_train_text, X_test_text,  y_train_lstm, y_test_lstm = train_test_split(
    X_text_padded, y, test_size=0.2, random_state=42, stratify=y
)
y_train_le_lstm=le.fit_transform(y_train_lstm)
y_test_le_lstm=le.fit_transform(y_test_lstm)
model = Sequential([
    Embedding(input_dim=5000, output_dim=128, input_length=max_sequence_length),
    LSTM(128, return_sequences=False),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(len(le.classes_), activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train_text, y_train_le_lstm, validation_data=(X_test_text, y_test_le_lstm), epochs=5, batch_size=32)

y_pred_lstm = model.predict(X_test_text)
y_pred_lstm_classes = y_pred_lstm.argmax(axis=1)
print("LSTM Results:")
print(classification_report(y_test_le_lstm, y_pred_lstm_classes))
```

```
Epoch 1/5
1317/1317 [==============================] - 96s 71ms/step - loss: 1.1
606 - accuracy: 0.5583 - val_loss: 0.9571 - val_accuracy: 0.6082
Epoch 2/5
1317/1317 [==============================] - 94s 71ms/step - loss: 0.9
162 - accuracy: 0.6569 - val_loss: 0.8823 - val_accuracy: 0.6750
Epoch 3/5
1317/1317 [==============================] - 101s 76ms/step - loss: 0.
7880 - accuracy: 0.7186 - val_loss: 0.8032 - val_accuracy: 0.7170
Epoch 4/5
1317/1317 [==============================] - 95s 72ms/step - loss: 0.6
856 - accuracy: 0.7594 - val_loss: 0.7677 - val_accuracy: 0.7253
Epoch 5/5
1317/1317 [==============================] - 95s 72ms/step - loss: 0.6
069 - accuracy: 0.7884 - val_loss: 0.7622 - val_accuracy: 0.7295
330/330 [==============================] - 9s 25ms/step
LSTM Results:
              precision    recall  f1-score   support

           0       0.65      0.80      0.72       768
           1       0.77      0.69      0.73       556
           2       0.70      0.61      0.66      3081
           3       0.90      0.90      0.90      3269
           4       0.75      0.33      0.45       215
           5       0.44      0.49      0.47       517
           6       0.62      0.73      0.67      2131

    accuracy                           0.73     10537
   macro avg       0.69      0.65      0.66     10537
weighted avg       0.74      0.73      0.73     10537
```

BERT + XGBoost

Machine Learning Pipeline

The pipeline consists of:

1. Sampling & Balancing the Data (to avoid class imbalance issues)
2. Embedding Text using BERT (to convert text into meaningful numerical representations)
3. Training an XGBoost Classifier (a powerful gradient boosting algorithm)

#

In [70]:
```
import tensorflow_hub as hub
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report

from sklearn.utils import resample
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
from sklearn.utils import resample
import os
os.environ['TF_USE_LEGACY_KERAS'] = '1'
import warnings
warnings.filterwarnings('ignore')
```

In [71]:
```
df.shape
```

Out[71]:
```
(52681, 4)
```

In [72]:
```
label_ordered = df['status'].value_counts().index

label_ordered = {k:i for i,k in enumerate(label_ordered,0)}

label_ordered
```

Out[72]:
```
{'Normal': 0,
 'Depression': 1,
 'Suicidal': 2,
 'Anxiety': 3,
 'Bipolar': 4,
 'Stress': 5,
 'Personality disorder': 6}
```

In [73]:
```
df['status'] = df['status'].map(label_ordered)
df.head()
```

Out[73]:

	status	cleaned_statement	num_of_characters	num_of_words
0	3	oh gosh	10	3
1	3	trouble sleeping confused mind restless heart ...	64	10
2	3	wrong back dear forward doubt stay restless re...	78	14
3	3	shifted focus something still worried	61	11
4	3	restless restless month boy mean	72	14

In [74]:
```
print(df['status'].value_counts())
```

```
status
0    16343
1    15404
2    10652
3     3841
4     2777
5     2587
6     1077
Name: count, dtype: int64
```

If we train a model on this dataset:

- The model might always predict the majority class (0: Legitimate) because it appears more frequently.
- Accuracy might be high, but the model fails to detect fraudulent cases (1).
- This leads to a biased and unreliable model

Step Process Purpose 1️⃣ Identify Class Distribution

- Count the number of samples in each class
- Detect imbalance

2️⃣ Resample Data

- Apply oversampling or undersampling
- Balance the dataset

3️⃣ Train Model

- Use the balanced dataset for training
- Improve accuracy & fairness

In [75]:
```
df_sample = df.sample(n=20000,random_state=2024)
majority_size = len(df_sample[df_sample['status']==0])

majority_size
```

Out[75]:
```
6140
```

In [76]:
```
from sklearn.utils import resample

def resampling_dfs(df):
    resample_minority = resample(df, replace=True, n_samples=majority_siz
e, random_state= 768)
    return resample_minority
```

In [77]:
```
df_sample
```

Out[77]:

	status	cleaned_statement	num_of_characters	num_of_words
43491	0	caitlinaudrey awww suck going sydney one	60	11
46017	4	feel like killing still one listens convinced ...	1355	259
24191	2	dl hellome hellodl hello know saydl okay dl so...	642	101
6893	0	body ache	9	2
40455	1	feel like believe really going trial run see g...	150	37
...	...	...	...	...
45045	0	argh stop yawning	22	4
17141	2	people love say suicide selfish way around sel...	229	44
42853	0	riskyrevenge wait sick	27	4
40028	1	long story sorry ex broke year ago remained fr...	7747	1726
9286	2	suffering must continue way	45	9

20000 rows × 4 columns

In [78]:
```
df_sample_0 = resampling_dfs(df_sample[df_sample['status']==0])
df_sample_1 = resampling_dfs(df_sample[df_sample['status']==1])
df_sample_2 = resampling_dfs(df_sample[df_sample['status']==2])
df_sample_3 = resampling_dfs(df_sample[df_sample['status']==3])
df_sample_4 = resampling_dfs(df_sample[df_sample['status']==4])
df_sample_5 = resampling_dfs(df_sample[df_sample['status']==5])
df_sample_6 = resampling_dfs(df_sample[df_sample['status']==6])

df_new =pd.concat([df_sample_0,df_sample_1,df_sample_2,df_sample_3,df_sam
ple_4,df_sample_5,df_sample_6], axis=0).reset_index(drop=True)

df_new['status'].value_counts()
```

Out[78]:
```
status
0    6140
1    6140
2    6140
3    6140
4    6140
5    6140
6    6140
Name: count, dtype: int64
```

In [79]:

```
df_new
```

Out[79]:

	status	cleaned_statement	num_of_characters	num_of_words
0	0	secret achieving success success happen prepar...	127	21
1	0	proud	10	2
2	0	seen far couple lizards	45	10
3	0	tell	31	6
4	0	welcome twitter	28	4
...	...	...	...	...
42975	6	interview anxiety job searching since august o...	508	91
42976	6	trying make friend someone lot friends making ...	1216	238
42977	6	please hear success stories yall got share	70	13
42978	6	also want get rid avpd avoiding makes much sen...	398	73
42979	6	hobbies comfortable sharing anyone hobbiesacti...	105	15

42980 rows × 4 columns

In [81]:

```
print("TF version:", tf.__version__)    # Should be 2.9, 2.10, 2.11, etc.
print("Hub version:", hub.__version__)  # ~0.12 or newer
print("Text version:", text.__version__)
```

```
TF version: 2.9.0
Hub version: 0.16.1
Text version: 2.9.0
```

Why Use BERT for Text Embeddings?

- Machine learning models can't understand text directly. They need numerical representations.
- Traditional techniques (like TF-IDF or Word2Vec) have limitations in understanding context.
- BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model developed by Google that:
  - Understands contextual meaning by looking at words in both directions (left and right).
  - Is pre-trained on massive text corpora (Wikipedia, books, etc.), so it understands language better than older models.

What is Happening Here?

1. ***bert_preprocess:***
   - Handles tokenization, lowercasing, and adding special tokens (like [CLS] for classification tasks).
   - Converts text into a format BERT understands.
2. ***bert_encoder:***
   - Uses the BERT model to generate embeddings.
   - Outputs 768-dimensional feature vectors for each input sentence.

In [82]:
```
bert_preprocess = hub.KerasLayer(
    "https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3",
    name="bert_preprocess_layer"
)
bert_encoder = hub.KerasLayer(
    "https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4",
    trainable=True,
    name="bert_encoder_layer"
)
```

- Takes raw text input (text_input).
- Uses BERT preprocessing (bert_preprocess) to prepare text.
- Passes the preprocessed text to BERT encoder (bert_encoder).
- Outputs pooled BERT embeddings (pooled_output), which are 768-dimensional vector representations of the entire sentence.

BERT provides two types of outputs:

1. sequence_output → Word-level embeddings (useful for Named Entity Recognition).
2. pooled_output → Sentence-level embeddings (useful for classification tasks).

📌 Since we are doing text classification, we use pooled_output, which represents the entire sentence.

In [84]:

```
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
encoder_outputs = bert_encoder(preprocessed_text)
pooled_output = encoder_outputs["pooled_output"]
embedding_model = tf.keras.Model(inputs=text_input, outputs=pooled_output)
```

- The dataset is split into training (X_train) and testing (X_test) sets.
- The text data is passed through the BERT model (embedding_model.predict()).
- BERT converts each text statement into a 768-dimensional vector.
- These vectors (train_embeddings and test_embeddings) are now ready for classification.

In [85]:

```
X_tr, X_te, y_tr, y_te = train_test_split(df_new['cleaned_statement'], df_new['status'], test_size=0.2, random_state=42)

train_embeddings = embedding_model.predict(X_tr)
test_embeddings = embedding_model.predict(X_te)
```

```
1075/1075 [==============================] - 8678s 8s/step
269/269 [==============================] - 2158s 8s/step
```

Why Do We Use BERT Before XGBoost?

- XGBoost does not understand text; it requires numerical input.
- BERT converts text into numerical embeddings, making it possible to train XGBoost.

📌 Without BERT, we would have to use TF-IDF or Word2Vec, which do not capture deep contextual meaning like BERT does.

❓ Why Use XGBoost for Text Classification?

- XGBoost (Extreme Gradient Boosting) is one of the most powerful tree-based machine learning models.
- It is used for structured data and tabular data.
- Better than traditional ML models like Logistic Regression and SVM because:
- Handles non-linearity and feature interactions automatically.
- Uses gradient boosting, meaning it learns iteratively and corrects mistakes at each step.
- Optimized for speed and handles missing values well.

In [86]:

```
xgb = XGBClassifier(alpha=0.5, lambda_=1.0,learning_rate=0.05, n_estimato
rs=500)
model = xgb.fit(train_embeddings, y_tr,
                eval_set=[(test_embeddings, y_te)], early_stopping_rounds
=10)
y_pred = model.predict(test_embeddings)
```

```
[0]	validation_0-mlogloss:1.89795
[1]	validation_0-mlogloss:1.85467
[2]	validation_0-mlogloss:1.81487
[3]	validation_0-mlogloss:1.77756
[4]	validation_0-mlogloss:1.74261
[5]	validation_0-mlogloss:1.70995
[6]	validation_0-mlogloss:1.67858
[7]	validation_0-mlogloss:1.64872
[8]	validation_0-mlogloss:1.62083
[9]	validation_0-mlogloss:1.59382
[10]	validation_0-mlogloss:1.56790
[11]	validation_0-mlogloss:1.54290
[12]	validation_0-mlogloss:1.51918
[13]	validation_0-mlogloss:1.49620
[14]	validation_0-mlogloss:1.47457
[15]	validation_0-mlogloss:1.45311
[16]	validation_0-mlogloss:1.43220
[17]	validation_0-mlogloss:1.41253
[18]	validation_0-mlogloss:1.39326
[19]	validation_0-mlogloss:1.37499
[20]	validation_0-mlogloss:1.35699
[21]	validation_0-mlogloss:1.33946
[22]	validation_0-mlogloss:1.32300
[23]	validation_0-mlogloss:1.30727
[24]	validation_0-mlogloss:1.29168
[25]	validation_0-mlogloss:1.27706
[26]	validation_0-mlogloss:1.26247
[27]	validation_0-mlogloss:1.24839
[28]	validation_0-mlogloss:1.23505
[29]	validation_0-mlogloss:1.22219
[30]	validation_0-mlogloss:1.20946
[31]	validation_0-mlogloss:1.19633
[32]	validation_0-mlogloss:1.18413
[33]	validation_0-mlogloss:1.17188
[34]	validation_0-mlogloss:1.16053
[35]	validation_0-mlogloss:1.14855
[36]	validation_0-mlogloss:1.13727
[37]	validation_0-mlogloss:1.12637
[38]	validation_0-mlogloss:1.11577
[39]	validation_0-mlogloss:1.10489
```

```
[40]	validation_0-mlogloss:1.09451
[41]	validation_0-mlogloss:1.08502
[42]	validation_0-mlogloss:1.07459
[43]	validation_0-mlogloss:1.06540
[44]	validation_0-mlogloss:1.05571
[45]	validation_0-mlogloss:1.04653
[46]	validation_0-mlogloss:1.03731
[47]	validation_0-mlogloss:1.02811
[48]	validation_0-mlogloss:1.01941
[49]	validation_0-mlogloss:1.01075
[50]	validation_0-mlogloss:1.00278
[51]	validation_0-mlogloss:0.99503
[52]	validation_0-mlogloss:0.98678
[53]	validation_0-mlogloss:0.97898
[54]	validation_0-mlogloss:0.97143
[55]	validation_0-mlogloss:0.96389
[56]	validation_0-mlogloss:0.95615
[57]	validation_0-mlogloss:0.94875
[58]	validation_0-mlogloss:0.94136
[59]	validation_0-mlogloss:0.93415
[60]	validation_0-mlogloss:0.92730
[61]	validation_0-mlogloss:0.91985
[62]	validation_0-mlogloss:0.91306
[63]	validation_0-mlogloss:0.90678
[64]	validation_0-mlogloss:0.90008
[65]	validation_0-mlogloss:0.89428
[66]	validation_0-mlogloss:0.88831
[67]	validation_0-mlogloss:0.88123
[68]	validation_0-mlogloss:0.87478
[69]	validation_0-mlogloss:0.86884
[70]	validation_0-mlogloss:0.86337
[71]	validation_0-mlogloss:0.85780
[72]	validation_0-mlogloss:0.85209
[73]	validation_0-mlogloss:0.84582
[74]	validation_0-mlogloss:0.83988
[75]	validation_0-mlogloss:0.83448
[76]	validation_0-mlogloss:0.82917
[77]	validation_0-mlogloss:0.82408
[78]	validation_0-mlogloss:0.81923
[79]	validation_0-mlogloss:0.81426
[80]	validation_0-mlogloss:0.80901
```

```
[81]	validation_0-mlogloss:0.80326
[82]	validation_0-mlogloss:0.79835
[83]	validation_0-mlogloss:0.79326
[84]	validation_0-mlogloss:0.78793
[85]	validation_0-mlogloss:0.78335
[86]	validation_0-mlogloss:0.77830
[87]	validation_0-mlogloss:0.77355
[88]	validation_0-mlogloss:0.76864
[89]	validation_0-mlogloss:0.76367
[90]	validation_0-mlogloss:0.75953
[91]	validation_0-mlogloss:0.75520
[92]	validation_0-mlogloss:0.75073
[93]	validation_0-mlogloss:0.74646
[94]	validation_0-mlogloss:0.74243
[95]	validation_0-mlogloss:0.73825
[96]	validation_0-mlogloss:0.73404
[97]	validation_0-mlogloss:0.72978
[98]	validation_0-mlogloss:0.72573
[99]	validation_0-mlogloss:0.72088
[100]	validation_0-mlogloss:0.71687
[101]	validation_0-mlogloss:0.71277
[102]	validation_0-mlogloss:0.70898
[103]	validation_0-mlogloss:0.70529
[104]	validation_0-mlogloss:0.70177
[105]	validation_0-mlogloss:0.69798
[106]	validation_0-mlogloss:0.69425
[107]	validation_0-mlogloss:0.69107
[108]	validation_0-mlogloss:0.68754
[109]	validation_0-mlogloss:0.68382
[110]	validation_0-mlogloss:0.68060
[111]	validation_0-mlogloss:0.67700
[112]	validation_0-mlogloss:0.67305
[113]	validation_0-mlogloss:0.66926
[114]	validation_0-mlogloss:0.66580
[115]	validation_0-mlogloss:0.66243
[116]	validation_0-mlogloss:0.65916
[117]	validation_0-mlogloss:0.65588
[118]	validation_0-mlogloss:0.65289
[119]	validation_0-mlogloss:0.65002
[120]	validation_0-mlogloss:0.64633
[121]	validation_0-mlogloss:0.64290
```

```
[122]	validation_0-mlogloss:0.63978
[123]	validation_0-mlogloss:0.63665
[124]	validation_0-mlogloss:0.63369
[125]	validation_0-mlogloss:0.63059
[126]	validation_0-mlogloss:0.62698
[127]	validation_0-mlogloss:0.62409
[128]	validation_0-mlogloss:0.62141
[129]	validation_0-mlogloss:0.61791
[130]	validation_0-mlogloss:0.61518
[131]	validation_0-mlogloss:0.61208
[132]	validation_0-mlogloss:0.60930
[133]	validation_0-mlogloss:0.60669
[134]	validation_0-mlogloss:0.60394
[135]	validation_0-mlogloss:0.60134
[136]	validation_0-mlogloss:0.59877
[137]	validation_0-mlogloss:0.59590
[138]	validation_0-mlogloss:0.59307
[139]	validation_0-mlogloss:0.59050
[140]	validation_0-mlogloss:0.58784
[141]	validation_0-mlogloss:0.58494
[142]	validation_0-mlogloss:0.58230
[143]	validation_0-mlogloss:0.57982
[144]	validation_0-mlogloss:0.57701
[145]	validation_0-mlogloss:0.57464
[146]	validation_0-mlogloss:0.57182
[147]	validation_0-mlogloss:0.56908
[148]	validation_0-mlogloss:0.56677
[149]	validation_0-mlogloss:0.56412
[150]	validation_0-mlogloss:0.56161
[151]	validation_0-mlogloss:0.55896
[152]	validation_0-mlogloss:0.55699
[153]	validation_0-mlogloss:0.55440
[154]	validation_0-mlogloss:0.55206
[155]	validation_0-mlogloss:0.54983
[156]	validation_0-mlogloss:0.54790
[157]	validation_0-mlogloss:0.54563
[158]	validation_0-mlogloss:0.54335
[159]	validation_0-mlogloss:0.54097
[160]	validation_0-mlogloss:0.53868
[161]	validation_0-mlogloss:0.53655
[162]	validation_0-mlogloss:0.53436
```

```
[163]	validation_0-mlogloss:0.53207
[164]	validation_0-mlogloss:0.52970
[165]	validation_0-mlogloss:0.52722
[166]	validation_0-mlogloss:0.52493
[167]	validation_0-mlogloss:0.52290
[168]	validation_0-mlogloss:0.52093
[169]	validation_0-mlogloss:0.51872
[170]	validation_0-mlogloss:0.51622
[171]	validation_0-mlogloss:0.51403
[172]	validation_0-mlogloss:0.51205
[173]	validation_0-mlogloss:0.51002
[174]	validation_0-mlogloss:0.50793
[175]	validation_0-mlogloss:0.50548
[176]	validation_0-mlogloss:0.50337
[177]	validation_0-mlogloss:0.50108
[178]	validation_0-mlogloss:0.49906
[179]	validation_0-mlogloss:0.49675
[180]	validation_0-mlogloss:0.49481
[181]	validation_0-mlogloss:0.49251
[182]	validation_0-mlogloss:0.49055
[183]	validation_0-mlogloss:0.48881
[184]	validation_0-mlogloss:0.48659
[185]	validation_0-mlogloss:0.48507
[186]	validation_0-mlogloss:0.48287
[187]	validation_0-mlogloss:0.48121
[188]	validation_0-mlogloss:0.47914
[189]	validation_0-mlogloss:0.47717
[190]	validation_0-mlogloss:0.47491
[191]	validation_0-mlogloss:0.47289
[192]	validation_0-mlogloss:0.47068
[193]	validation_0-mlogloss:0.46888
[194]	validation_0-mlogloss:0.46718
[195]	validation_0-mlogloss:0.46485
[196]	validation_0-mlogloss:0.46276
[197]	validation_0-mlogloss:0.46067
[198]	validation_0-mlogloss:0.45878
[199]	validation_0-mlogloss:0.45706
[200]	validation_0-mlogloss:0.45532
[201]	validation_0-mlogloss:0.45367
[202]	validation_0-mlogloss:0.45179
[203]	validation_0-mlogloss:0.44999
```

```
[204]	validation_0-mlogloss:0.44830
[205]	validation_0-mlogloss:0.44652
[206]	validation_0-mlogloss:0.44453
[207]	validation_0-mlogloss:0.44295
[208]	validation_0-mlogloss:0.44104
[209]	validation_0-mlogloss:0.43935
[210]	validation_0-mlogloss:0.43781
[211]	validation_0-mlogloss:0.43614
[212]	validation_0-mlogloss:0.43461
[213]	validation_0-mlogloss:0.43286
[214]	validation_0-mlogloss:0.43123
[215]	validation_0-mlogloss:0.42917
[216]	validation_0-mlogloss:0.42722
[217]	validation_0-mlogloss:0.42596
[218]	validation_0-mlogloss:0.42428
[219]	validation_0-mlogloss:0.42272
[220]	validation_0-mlogloss:0.42149
[221]	validation_0-mlogloss:0.42002
[222]	validation_0-mlogloss:0.41840
[223]	validation_0-mlogloss:0.41692
[224]	validation_0-mlogloss:0.41517
[225]	validation_0-mlogloss:0.41361
[226]	validation_0-mlogloss:0.41194
[227]	validation_0-mlogloss:0.41072
[228]	validation_0-mlogloss:0.40921
[229]	validation_0-mlogloss:0.40756
[230]	validation_0-mlogloss:0.40601
[231]	validation_0-mlogloss:0.40458
[232]	validation_0-mlogloss:0.40316
[233]	validation_0-mlogloss:0.40180
[234]	validation_0-mlogloss:0.40032
[235]	validation_0-mlogloss:0.39884
[236]	validation_0-mlogloss:0.39727
[237]	validation_0-mlogloss:0.39573
[238]	validation_0-mlogloss:0.39413
[239]	validation_0-mlogloss:0.39293
[240]	validation_0-mlogloss:0.39167
[241]	validation_0-mlogloss:0.39038
[242]	validation_0-mlogloss:0.38911
[243]	validation_0-mlogloss:0.38781
[244]	validation_0-mlogloss:0.38646
```

```
[245]	validation_0-mlogloss:0.38529
[246]	validation_0-mlogloss:0.38385
[247]	validation_0-mlogloss:0.38250
[248]	validation_0-mlogloss:0.38126
[249]	validation_0-mlogloss:0.37978
[250]	validation_0-mlogloss:0.37870
[251]	validation_0-mlogloss:0.37732
[252]	validation_0-mlogloss:0.37592
[253]	validation_0-mlogloss:0.37462
[254]	validation_0-mlogloss:0.37328
[255]	validation_0-mlogloss:0.37203
[256]	validation_0-mlogloss:0.37069
[257]	validation_0-mlogloss:0.36957
[258]	validation_0-mlogloss:0.36818
[259]	validation_0-mlogloss:0.36698
[260]	validation_0-mlogloss:0.36590
[261]	validation_0-mlogloss:0.36463
[262]	validation_0-mlogloss:0.36337
[263]	validation_0-mlogloss:0.36235
[264]	validation_0-mlogloss:0.36124
[265]	validation_0-mlogloss:0.36004
[266]	validation_0-mlogloss:0.35900
[267]	validation_0-mlogloss:0.35767
[268]	validation_0-mlogloss:0.35657
[269]	validation_0-mlogloss:0.35544
[270]	validation_0-mlogloss:0.35415
[271]	validation_0-mlogloss:0.35306
[272]	validation_0-mlogloss:0.35194
[273]	validation_0-mlogloss:0.35078
[274]	validation_0-mlogloss:0.34951
[275]	validation_0-mlogloss:0.34816
[276]	validation_0-mlogloss:0.34705
[277]	validation_0-mlogloss:0.34584
[278]	validation_0-mlogloss:0.34482
[279]	validation_0-mlogloss:0.34352
[280]	validation_0-mlogloss:0.34249
[281]	validation_0-mlogloss:0.34159
[282]	validation_0-mlogloss:0.34027
[283]	validation_0-mlogloss:0.33911
[284]	validation_0-mlogloss:0.33804
[285]	validation_0-mlogloss:0.33711
```

```
[286]	validation_0-mlogloss:0.33624
[287]	validation_0-mlogloss:0.33518
[288]	validation_0-mlogloss:0.33405
[289]	validation_0-mlogloss:0.33296
[290]	validation_0-mlogloss:0.33183
[291]	validation_0-mlogloss:0.33087
[292]	validation_0-mlogloss:0.32983
[293]	validation_0-mlogloss:0.32902
[294]	validation_0-mlogloss:0.32792
[295]	validation_0-mlogloss:0.32693
[296]	validation_0-mlogloss:0.32582
[297]	validation_0-mlogloss:0.32482
[298]	validation_0-mlogloss:0.32390
[299]	validation_0-mlogloss:0.32296
[300]	validation_0-mlogloss:0.32210
[301]	validation_0-mlogloss:0.32133
[302]	validation_0-mlogloss:0.32037
[303]	validation_0-mlogloss:0.31952
[304]	validation_0-mlogloss:0.31864
[305]	validation_0-mlogloss:0.31785
[306]	validation_0-mlogloss:0.31691
[307]	validation_0-mlogloss:0.31584
[308]	validation_0-mlogloss:0.31481
[309]	validation_0-mlogloss:0.31389
[310]	validation_0-mlogloss:0.31298
[311]	validation_0-mlogloss:0.31194
[312]	validation_0-mlogloss:0.31107
[313]	validation_0-mlogloss:0.31015
[314]	validation_0-mlogloss:0.30922
[315]	validation_0-mlogloss:0.30828
[316]	validation_0-mlogloss:0.30736
[317]	validation_0-mlogloss:0.30642
[318]	validation_0-mlogloss:0.30565
[319]	validation_0-mlogloss:0.30496
[320]	validation_0-mlogloss:0.30410
[321]	validation_0-mlogloss:0.30334
[322]	validation_0-mlogloss:0.30251
[323]	validation_0-mlogloss:0.30163
[324]	validation_0-mlogloss:0.30072
[325]	validation_0-mlogloss:0.29981
[326]	validation_0-mlogloss:0.29891
```

```
[327]	validation_0-mlogloss:0.29809
[328]	validation_0-mlogloss:0.29720
[329]	validation_0-mlogloss:0.29635
[330]	validation_0-mlogloss:0.29553
[331]	validation_0-mlogloss:0.29461
[332]	validation_0-mlogloss:0.29383
[333]	validation_0-mlogloss:0.29296
[334]	validation_0-mlogloss:0.29212
[335]	validation_0-mlogloss:0.29133
[336]	validation_0-mlogloss:0.29048
[337]	validation_0-mlogloss:0.28971
[338]	validation_0-mlogloss:0.28897
[339]	validation_0-mlogloss:0.28805
[340]	validation_0-mlogloss:0.28720
[341]	validation_0-mlogloss:0.28645
[342]	validation_0-mlogloss:0.28573
[343]	validation_0-mlogloss:0.28493
[344]	validation_0-mlogloss:0.28423
[345]	validation_0-mlogloss:0.28358
[346]	validation_0-mlogloss:0.28284
[347]	validation_0-mlogloss:0.28217
[348]	validation_0-mlogloss:0.28143
[349]	validation_0-mlogloss:0.28067
[350]	validation_0-mlogloss:0.27999
[351]	validation_0-mlogloss:0.27929
[352]	validation_0-mlogloss:0.27851
[353]	validation_0-mlogloss:0.27789
[354]	validation_0-mlogloss:0.27719
[355]	validation_0-mlogloss:0.27649
[356]	validation_0-mlogloss:0.27586
[357]	validation_0-mlogloss:0.27517
[358]	validation_0-mlogloss:0.27457
[359]	validation_0-mlogloss:0.27386
[360]	validation_0-mlogloss:0.27303
[361]	validation_0-mlogloss:0.27241
[362]	validation_0-mlogloss:0.27186
[363]	validation_0-mlogloss:0.27125
[364]	validation_0-mlogloss:0.27074
[365]	validation_0-mlogloss:0.27020
[366]	validation_0-mlogloss:0.26949
[367]	validation_0-mlogloss:0.26877
```

```
[368]	validation_0-mlogloss:0.26805
[369]	validation_0-mlogloss:0.26745
[370]	validation_0-mlogloss:0.26675
[371]	validation_0-mlogloss:0.26609
[372]	validation_0-mlogloss:0.26543
[373]	validation_0-mlogloss:0.26473
[374]	validation_0-mlogloss:0.26406
[375]	validation_0-mlogloss:0.26347
[376]	validation_0-mlogloss:0.26288
[377]	validation_0-mlogloss:0.26224
[378]	validation_0-mlogloss:0.26159
[379]	validation_0-mlogloss:0.26098
[380]	validation_0-mlogloss:0.26037
[381]	validation_0-mlogloss:0.25971
[382]	validation_0-mlogloss:0.25918
[383]	validation_0-mlogloss:0.25853
[384]	validation_0-mlogloss:0.25789
[385]	validation_0-mlogloss:0.25720
[386]	validation_0-mlogloss:0.25664
[387]	validation_0-mlogloss:0.25602
[388]	validation_0-mlogloss:0.25532
[389]	validation_0-mlogloss:0.25475
[390]	validation_0-mlogloss:0.25417
[391]	validation_0-mlogloss:0.25349
[392]	validation_0-mlogloss:0.25289
[393]	validation_0-mlogloss:0.25238
[394]	validation_0-mlogloss:0.25182
[395]	validation_0-mlogloss:0.25130
[396]	validation_0-mlogloss:0.25078
[397]	validation_0-mlogloss:0.25018
[398]	validation_0-mlogloss:0.24960
[399]	validation_0-mlogloss:0.24907
[400]	validation_0-mlogloss:0.24853
[401]	validation_0-mlogloss:0.24803
[402]	validation_0-mlogloss:0.24739
[403]	validation_0-mlogloss:0.24679
[404]	validation_0-mlogloss:0.24622
[405]	validation_0-mlogloss:0.24565
[406]	validation_0-mlogloss:0.24511
[407]	validation_0-mlogloss:0.24461
[408]	validation_0-mlogloss:0.24392
```

```
[409]	validation_0-mlogloss:0.24340
[410]	validation_0-mlogloss:0.24282
[411]	validation_0-mlogloss:0.24228
[412]	validation_0-mlogloss:0.24175
[413]	validation_0-mlogloss:0.24130
[414]	validation_0-mlogloss:0.24082
[415]	validation_0-mlogloss:0.24034
[416]	validation_0-mlogloss:0.23967
[417]	validation_0-mlogloss:0.23918
[418]	validation_0-mlogloss:0.23867
[419]	validation_0-mlogloss:0.23822
[420]	validation_0-mlogloss:0.23782
[421]	validation_0-mlogloss:0.23730
[422]	validation_0-mlogloss:0.23675
[423]	validation_0-mlogloss:0.23628
[424]	validation_0-mlogloss:0.23582
[425]	validation_0-mlogloss:0.23535
[426]	validation_0-mlogloss:0.23486
[427]	validation_0-mlogloss:0.23438
[428]	validation_0-mlogloss:0.23397
[429]	validation_0-mlogloss:0.23355
[430]	validation_0-mlogloss:0.23308
[431]	validation_0-mlogloss:0.23266
[432]	validation_0-mlogloss:0.23223
[433]	validation_0-mlogloss:0.23183
[434]	validation_0-mlogloss:0.23140
[435]	validation_0-mlogloss:0.23084
[436]	validation_0-mlogloss:0.23037
[437]	validation_0-mlogloss:0.22990
[438]	validation_0-mlogloss:0.22951
[439]	validation_0-mlogloss:0.22908
[440]	validation_0-mlogloss:0.22868
[441]	validation_0-mlogloss:0.22824
[442]	validation_0-mlogloss:0.22785
[443]	validation_0-mlogloss:0.22735
[444]	validation_0-mlogloss:0.22690
[445]	validation_0-mlogloss:0.22638
[446]	validation_0-mlogloss:0.22591
[447]	validation_0-mlogloss:0.22543
[448]	validation_0-mlogloss:0.22506
[449]	validation_0-mlogloss:0.22473
```

```
[450]	validation_0-mlogloss:0.22432
[451]	validation_0-mlogloss:0.22385
[452]	validation_0-mlogloss:0.22344
[453]	validation_0-mlogloss:0.22306
[454]	validation_0-mlogloss:0.22266
[455]	validation_0-mlogloss:0.22226
[456]	validation_0-mlogloss:0.22183
[457]	validation_0-mlogloss:0.22137
[458]	validation_0-mlogloss:0.22099
[459]	validation_0-mlogloss:0.22059
[460]	validation_0-mlogloss:0.22020
[461]	validation_0-mlogloss:0.21990
[462]	validation_0-mlogloss:0.21955
[463]	validation_0-mlogloss:0.21919
[464]	validation_0-mlogloss:0.21881
[465]	validation_0-mlogloss:0.21844
[466]	validation_0-mlogloss:0.21807
[467]	validation_0-mlogloss:0.21768
[468]	validation_0-mlogloss:0.21739
[469]	validation_0-mlogloss:0.21702
[470]	validation_0-mlogloss:0.21663
[471]	validation_0-mlogloss:0.21632
[472]	validation_0-mlogloss:0.21592
[473]	validation_0-mlogloss:0.21558
[474]	validation_0-mlogloss:0.21515
[475]	validation_0-mlogloss:0.21481
[476]	validation_0-mlogloss:0.21445
[477]	validation_0-mlogloss:0.21411
[478]	validation_0-mlogloss:0.21370
[479]	validation_0-mlogloss:0.21328
[480]	validation_0-mlogloss:0.21289
[481]	validation_0-mlogloss:0.21257
[482]	validation_0-mlogloss:0.21214
[483]	validation_0-mlogloss:0.21185
[484]	validation_0-mlogloss:0.21150
[485]	validation_0-mlogloss:0.21108
[486]	validation_0-mlogloss:0.21072
[487]	validation_0-mlogloss:0.21038
[488]	validation_0-mlogloss:0.21001
[489]	validation_0-mlogloss:0.20970
[490]	validation_0-mlogloss:0.20936
```

```
[491]	validation_0-mlogloss:0.20912
[492]	validation_0-mlogloss:0.20885
[493]	validation_0-mlogloss:0.20852
[494]	validation_0-mlogloss:0.20823
[495]	validation_0-mlogloss:0.20791
[496]	validation_0-mlogloss:0.20762
[497]	validation_0-mlogloss:0.20725
[498]	validation_0-mlogloss:0.20690
[499]	validation_0-mlogloss:0.20659
```

In [87]:

```
print("\nClassification Report:\n", classification_report(y_te, y_pred))
print(f"\n Test Accuracy: {accuracy_score(y_te, y_pred):.4f}")
```

```
🔹 Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      1260
           1       0.87      0.78      0.83      1220
           2       0.86      0.87      0.87      1187
           3       0.97      0.98      0.98      1252
           4       0.97      1.00      0.98      1215
           5       0.96      1.00      0.98      1210
           6       0.99      1.00      1.00      1252

    accuracy                           0.94      8596
   macro avg       0.94      0.94      0.94      8596
weighted avg       0.94      0.94      0.94      8596


✅ Test Accuracy: 0.9404
```