In [36]:
```
%pip install TextBlob
%pip install WordCloud
```

```
Requirement already satisfied: TextBlob in /opt/anaconda3/lib/python3.12/sit
e-packages (0.18.0.post0)
Requirement already satisfied: nltk>=3.8 in /opt/anaconda3/lib/python3.12/si
te-packages (from TextBlob) (3.8.1)
Requirement already satisfied: click in /opt/anaconda3/lib/python3.12/site-p
ackages (from nltk>=3.8->TextBlob) (8.1.7)
Requirement already satisfied: joblib in /opt/anaconda3/lib/python3.12/site-
packages (from nltk>=3.8->TextBlob) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /opt/anaconda3/lib/python
3.12/site-packages (from nltk>=3.8->TextBlob) (2023.10.3)
Requirement already satisfied: tqdm in /opt/anaconda3/lib/python3.12/site-pa
ckages (from nltk>=3.8->TextBlob) (4.66.4)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: WordCloud in /opt/anaconda3/lib/python3.12/si
te-packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in /opt/anaconda3/lib/python3.1
2/site-packages (from WordCloud) (1.26.4)
Requirement already satisfied: pillow in /opt/anaconda3/lib/python3.12/site-
packages (from WordCloud) (10.3.0)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.12/s
ite-packages (from WordCloud) (3.8.4)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib->WordCloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.1
2/site-packages (from matplotlib->WordCloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from matplotlib->WordCloud) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/anaconda3/lib/pytho
n3.12/site-packages (from matplotlib->WordCloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib->WordCloud) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python
3.12/site-packages (from matplotlib->WordCloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /opt/anaconda3/lib/py
thon3.12/site-packages (from matplotlib->WordCloud) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/sit
e-packages (from python-dateutil>=2.7->matplotlib->WordCloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [37]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accurac
import plotly.figure_factory as ff
```

```
from textblob import TextBlob
import numpy as np
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [38]:
```
# Load the data
path = './kaggle_sentiment_data.csv'
df = pd.read_csv(path)
```

In [39]:
```
# Display the first few rows of the dataframe
print(df.head())
```

```
   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 [40]:
```
# EDA
print("Dataset Info:")
print(df.info())
```

```
Dataset 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
None
```

In [41]:
```
print("Missing Values:")
print(df.isnull().sum())
```

```
Missing Values:
Unnamed: 0      0
statement     362
status          0
dtype: int64
```

In [42]:
```
# Distribution of target labels
fig = px.histogram(df, x='status', title='Distribution of Mental Health Stat
fig.show()
```

In [43]:
```
# Handle NaN values in the statement column
df['statement'] = df['statement'].fillna('')
```

In [44]:
```
# Text Length Distribution
df['text_length'] = df['statement'].apply(lambda x: len(str(x).split()))
```

```
fig = px.histogram(df, x='text_length', title='Text Length Distribution')
fig.show()
```

In [45]:
```
# Data Preprocessing
nltk.download('stopwords')
nltk.download('punkt')

def preprocess_text(text):
    text = text.lower()  # Lowercase text
    text = re.sub(r'\[.*?\]', '', text)  # Remove text in square brackets
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Remove links
    text = re.sub(r'<.*?>+', '', text)  # Remove HTML tags
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)  # Remo
    text = re.sub(r'\n', '', text)  # Remove newlines
    text = re.sub(r'\w*\d\w*', '', text)  # Remove words containing numbers
    return text

df['cleaned_statement'] = df['statement'].apply(lambda x: preprocess_text(x)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/vallipaladugu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/vallipaladugu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [46]:
```
# Tokenization and Stopwords Removal
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)

df['cleaned_statement'] = df['cleaned_statement'].apply(lambda x: remove_sto
```

In [47]:
```
# Data Augmentation
def augment_text(text):
    try:
        blob = TextBlob(text)
        translated = blob.translate(to='fr').translate(to='en')
        return str(translated)
    except Exception as e:
        return text

df['augmented_statement'] = df['statement'].apply(augment_text)
augmented_df = df[['statement', 'status']].copy()
augmented_df['statement'] = df['augmented_statement']
df = pd.concat([df, augmented_df])
```

In [48]:
```
# Reapply preprocessing on augmented data
df['cleaned_statement'] = df['statement'].apply(lambda x: preprocess_text(x)
df['cleaned_statement'] = df['cleaned_statement'].apply(lambda x: remove_sto
```

In [49]:
```
# Ensure no NaN values are left
df['cleaned_statement'] = df['cleaned_statement'].fillna('')
```

In [50]:
```
# Splitting the data
X = df['cleaned_statement']
y = df['status']
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```

In [52]:
```
# Vectorization
vectorizer = TfidfVectorizer(max_features=10000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [53]:
```
# Model Training with Hyperparameter Tuning
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100]
}

model = LogisticRegression(max_iter=1000)
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_tfidf, y_train)

# Best Model
best_model = grid_search.best_estimator_
```

In [54]:
```
# Predictions
y_pred = best_model.predict(X_test_tfidf)
```

In [55]:
```
# Evaluation
print("Best Parameters:")
print(grid_search.best_params_)

print("Accuracy Score:")
print(accuracy_score(y_test, y_pred))

print("Classification Report:")
print(classification_report(y_test, y_pred))
```

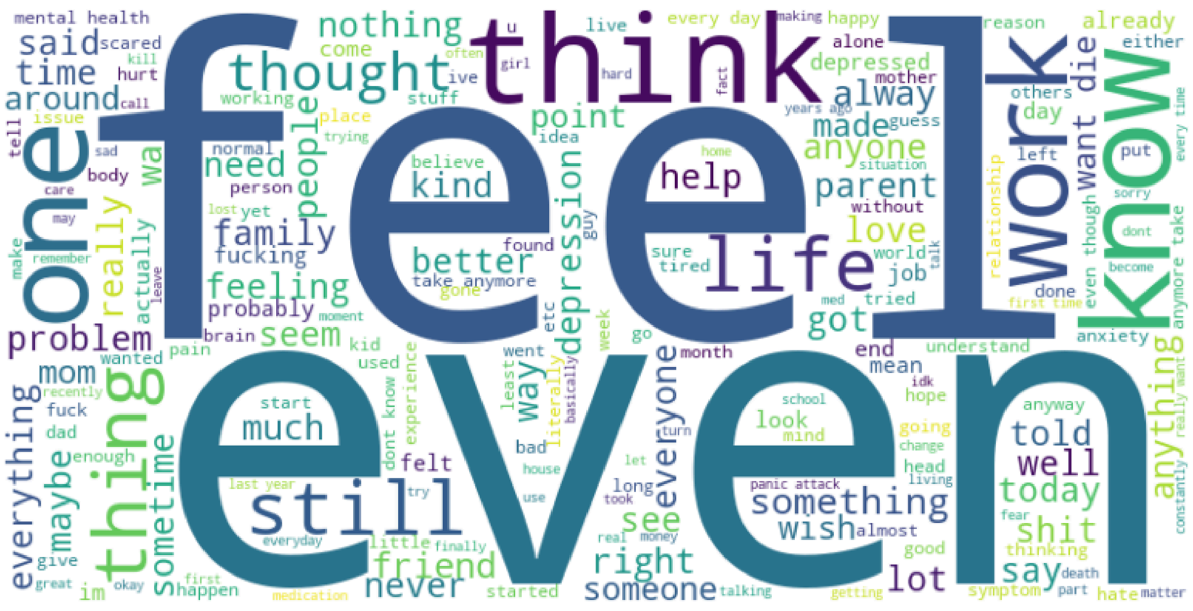
```
Best Parameters:
{'C': 100}
Accuracy Score:
0.8637477613347158
Classification Report:
                      precision    recall  f1-score   support

             Anxiety       0.92      0.91      0.91      1562
             Bipolar       0.93      0.90      0.92      1150
          Depression       0.83      0.82      0.82      6182
              Normal       0.93      0.96      0.94      6571
Personality disorder       0.85      0.81      0.83       447
              Stress       0.89      0.85      0.87      1047
            Suicidal       0.77      0.76      0.77      4259

            accuracy                           0.86     21218
           macro avg       0.87      0.86      0.87     21218
        weighted avg       0.86      0.86      0.86     21218
```

In [56]:
```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
cm_fig = ff.create_annotated_heatmap(
    z=cm,
    x=list(set(y_test)),
    y=list(set(y_test)),
    annotation_text=cm,
    colorscale='Viridis'
)
cm_fig.update_layout(title='Confusion Matrix')
cm_fig.update_layout(title='Confusion Matrix', width=800, height=600)
cm_fig.show()
```

In [57]:
```
# Feature Importance
feature_names = vectorizer.get_feature_names_out()
coefs = best_model.coef_
for i, category in enumerate(best_model.classes_):
    top_features = coefs[i].argsort()[-10:]
    top_words = [feature_names[j] for j in top_features]
    top_scores = [coefs[i][j] for j in top_features]
    fig = go.Figure([go.Bar(x=top_words, y=top_scores)])
    fig.update_layout(title=f'Top Features for {category}', width=800, heigh
    fig.show()
```

In [58]:
```
# Word Cloud
all_text = ' '.join(df['cleaned_statement'])
wordcloud = WordCloud(width=800, height=400, background_color='white').gener
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Cleaned Statements')
plt.show()
```

Word Cloud of Cleaned Statements



In [59]:
```
# Status Distribution
fig = px.pie(df, names='status', title='Proportion of Each Status Category')
fig.update_layout(width=800, height=600)
fig.show()
```