This notebook runs Naive Bayes Classifier with Faizan's preprocessing tools without the augmented text function.

Accuracy: 0.678386275803563

In [4]:

```
import pandas as pd
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, Stratifi
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report, accuracy_score, confusion
from sklearn.naive_bayes import MultinomialNB
import plotly.figure_factory as ff
```

In [5]:

```
# Load the data
path = './kaggle_sentiment_data.csv'
df = pd.read_csv(path)
```

In [6]:

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

In [7]:

```
# Data Preprocessing
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
```

In [8]:

```
# 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)
```

In [17]:

```
# Preprocess the text data
df['cleaned_statement'] = df['statement'].apply(preprocess_text).apply(remov

# Ensure no NaN values
df['cleaned_statement'] = df['cleaned_statement'].fillna('')

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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```

In [18]:
```
# Vectorization
vectorizer = TfidfVectorizer(max_features=10000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [19]:
```
# Define parameter grid for MultinomialNB
param_grid = {
    'alpha': [0, 0.0000001, 0.1, 0.5, 1.0, 2.0, 5.0],
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

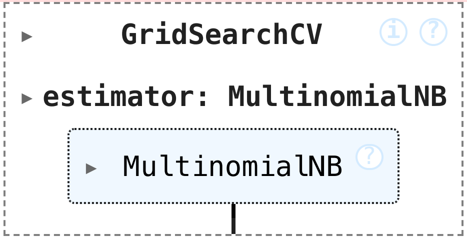
In [20]:
```
# GridSearchCV with MultinomialNB
grid_search = GridSearchCV(
    MultinomialNB(),
    param_grid,
    scoring='accuracy',
    cv=cv,
    verbose=1,
    n_jobs=-1
)

grid_search.fit(X_train_tfidf, y_train)
```

```
Fitting 5 folds for each of 7 candidates, totalling 35 fits
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/naive_bayes.py:890: Runt
imeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/naive_bayes.py:890: Runt
imeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/naive_bayes.py:890: Runt
imeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/naive_bayes.py:890: Runt
imeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/naive_bayes.py:890: Runt
imeWarning: divide by zero encountered in log
  self.feature_log_prob_ = np.log(smoothed_fc) - np.log(
```

Out[20]:



▸ **GridSearchCV**

▸ **estimator: MultinomialNB**

▸ MultinomialNB

In [21]:
```
# Get best model and hyperparameters
best_model = grid_search.best_estimator_
best_params = grid_search.best_params_
```

```
print("Best Parameters:", best_params)
```

```
Best Parameters: {'alpha': 0.1}
```

In [22]:

```
# Make predictions
y_pred = best_model.predict(X_test_tfidf)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.6736732962578943
                      precision    recall  f1-score   support

             Anxiety       0.78      0.64      0.71       778
             Bipolar       0.80      0.52      0.63       575
          Depression       0.53      0.79      0.63      3081
              Normal       0.85      0.80      0.82      3270
Personality disorder       0.93      0.21      0.35       240
              Stress       0.76      0.23      0.35       534
            Suicidal       0.67      0.54      0.60      2131

            accuracy                           0.67     10609
           macro avg       0.76      0.53      0.58     10609
        weighted avg       0.71      0.67      0.67     10609
```

In [15]:

```
# 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()
```