In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

Data Preprocessing

In this section, we clean and prepare the dataset for sentiment analysis.

In [6]:

```
df = pd.read_csv('Combined Data.csv')
```

In [7]:

```
df.head()
```

Out[7]:

	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 [11]:

```
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 [13]:

```
df.drop('Unnamed: 0', axis=1, inplace = True)
```

In [15]:

```
df.head()
```

Out[15]:

	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 [17]:

```
df.isnull().sum()
```

Out[17]:

```
statement    362
status         0
dtype: int64
```

In [19]:

```
most_frequent = df['statement'].mode()[0]
df['statement'].fillna(most_frequent,inplace=True)
```

```
C:\Users\negar\AppData\Local\Temp\ipykernel_900\1921006020.py:2: FutureWarning: A va
lue is trying to be set on a copy of a DataFrame or Series through chained assignmen
t using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['statement'].fillna(most_frequent,inplace=True)
```

In [21]:

```
df.isnull().sum()
```

Out[21]:

```
statement    0
status       0
dtype: int64
```

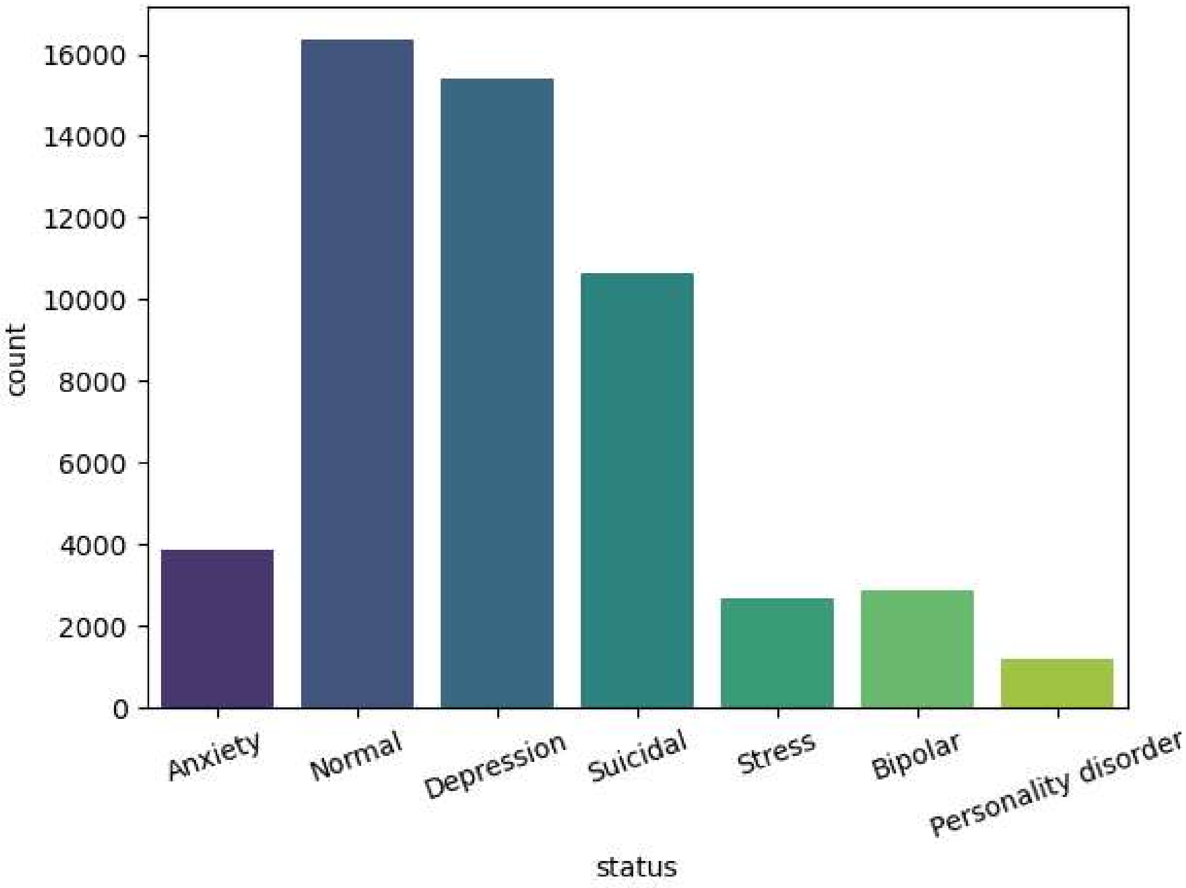
Exploratory Data Analysis (EDA)

Here, we analyze the dataset to understand its structure and distribution.

In [24]:

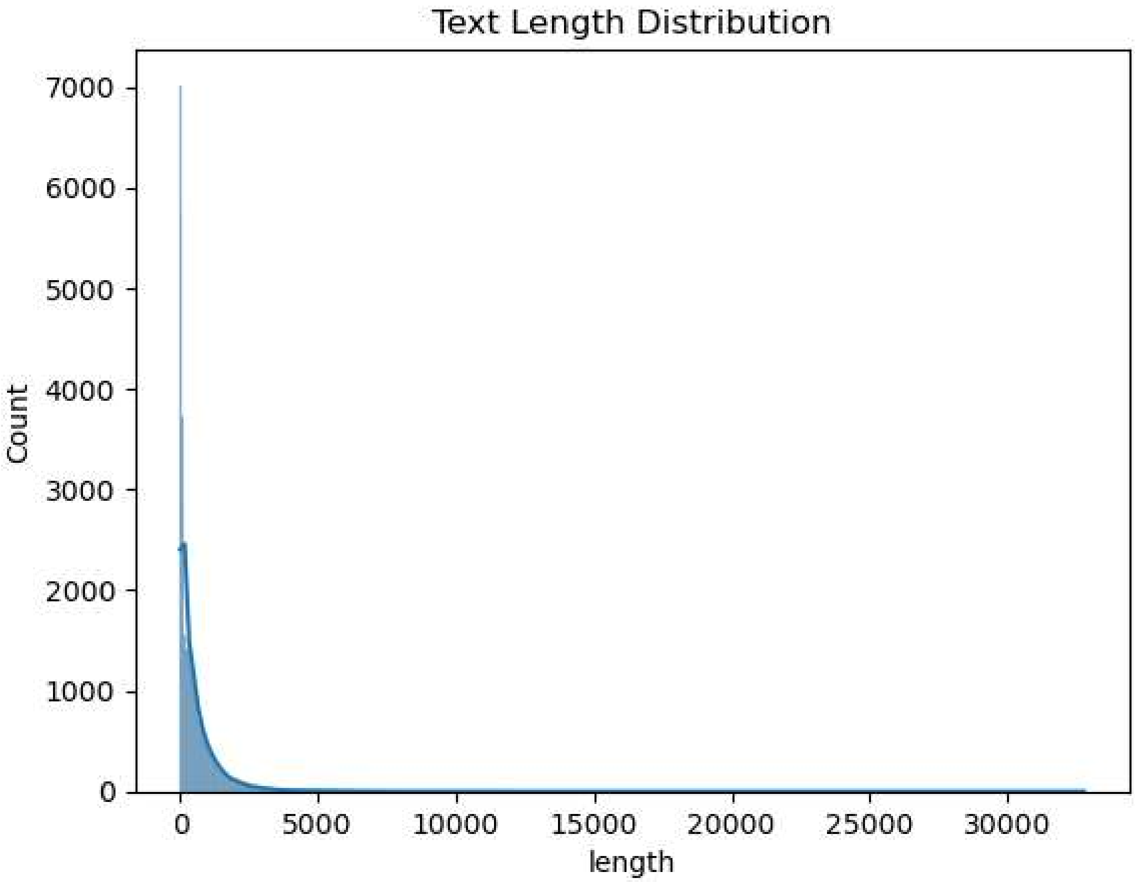
```
sns.countplot(df, x='status',palette='viridis')
plt.xticks(rotation=20)
plt.tight_layout()
```

```
C:\Users\negar\AppData\Local\Temp\ipykernel_900\1708889018.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(df, x='status',palette='viridis')
```



```
In [26]: df['length'] = df['statement'].apply(len)
         sns.histplot(df['length'], kde=True)
         plt.title('Text Length Distribution')
```

```
Out[26]: Text(0.5, 1.0, 'Text Length Distribution')
```

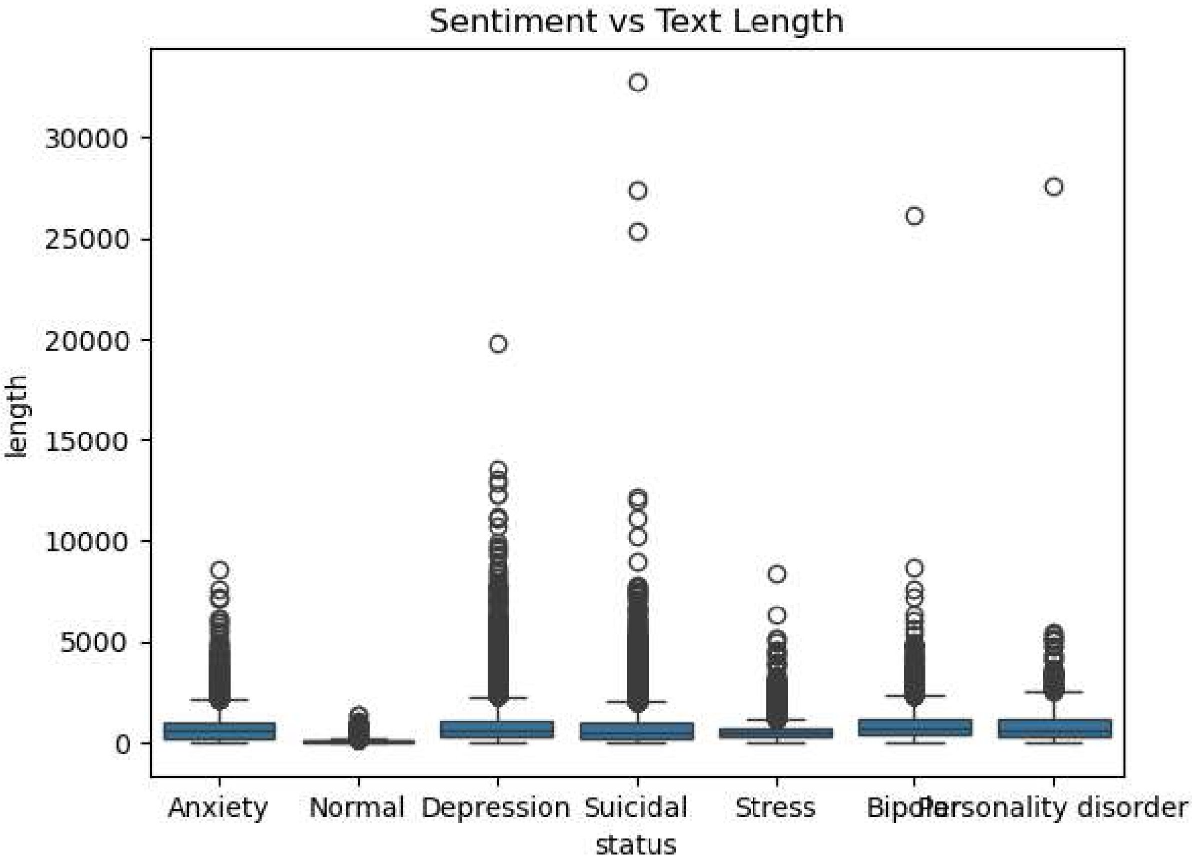


In [28]:

```
sns.boxplot(x='status', y='length', data=df)
plt.title('Sentiment vs Text Length')
```

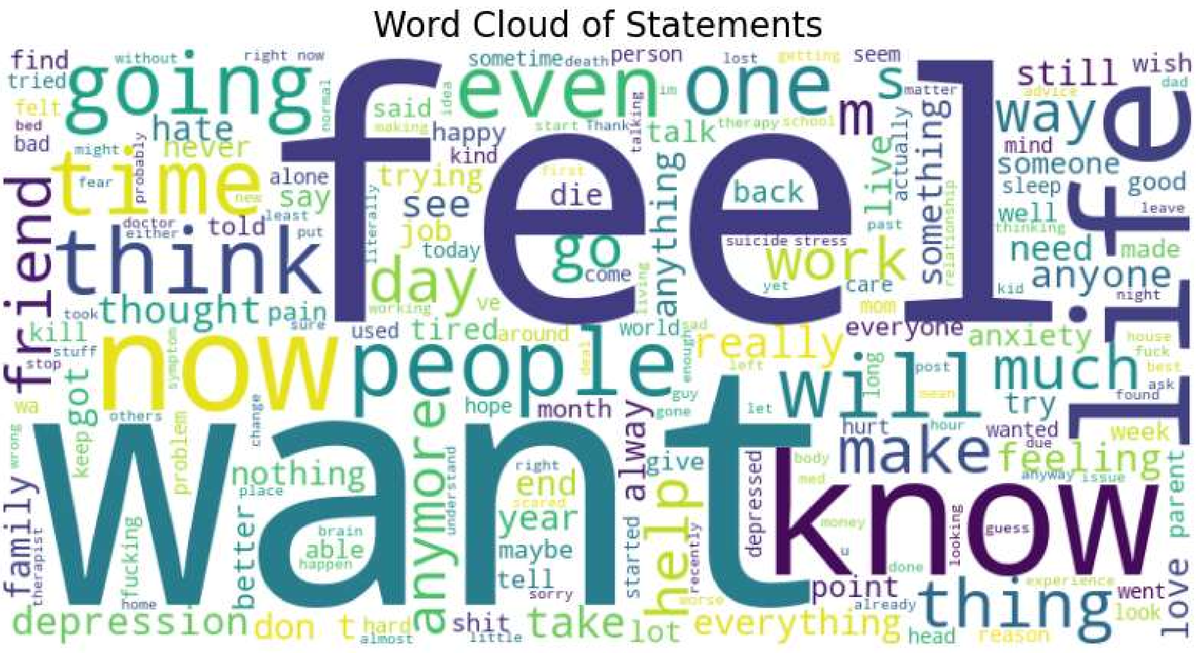
Out[28]:

```
Text(0.5, 1.0, 'Sentiment vs Text Length')
```



In [30]:
```python
text = " ".join(df['statement'].astype(str))
```

In [66]:
```python
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=400, background_color='white', colormap='vi

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Statements', fontsize=16)
plt.show()
```



Word Cloud of Statements

Training Models

In this section, we train various machine learning models for sentiment classification. We compare algorithms such as Logistic Regression, Decision Tress, and Neural Networks performance with metrics like accuracy, precision, recall, and F1-score help evaluate model effectiveness.

In [34]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [36]:
```
le = LabelEncoder()
```

In [38]:
```
df['status'] = le.fit_transform(df['status'])
```

In [40]:
```
X = df['statement']
y = df['status']
```

In [42]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

In [44]:
```
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

In [57]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
import lightgbm as lgb
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score
```

```
models = [
    LogisticRegression(multi_class='ovr', solver='lbfgs'),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    XGBClassifier(),
    lgb.LGBMClassifier(),
    MLPClassifier(hidden_layer_sizes=(100,), max_iter=500)

]
results = {}
for model in models:
    model_name = model.__class__.__name__
    model.fit(X_train_tfidf, y_train)

    y_pred = model.predict(X_test_tfidf)
    accuracy = accuracy_score(y_test, y_pred)
    class_report = classification_report(y_test, y_pred)
    results[model_name] = {
        "accuracy": accuracy,
        "classification_report": class_report
    }

for model_name, result in results.items():
    print(f"Model: {model_name}")
    print(f"Accuracy: {result['accuracy']}")
    print("Classification Report:")
    print(result['classification_report'])
    print("-" * 50)
```

```
C:\Users\negar\anaconda3\Lib\site-packages\joblib\externals\loky\backend\context.py:
136: UserWarning: Could not find the number of physical cores for the following reas
on:
[WinError 2] The system cannot find the file specified
Returning the number of logical cores instead. You can silence this warning by setti
ng LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File "C:\Users\negar\anaconda3\Lib\site-packages\joblib\externals\loky\backend\con
text.py", line 257, in _count_physical_cores
    cpu_info = subprocess.run(
               ^^^^^^^^^^^^^^^
  File "C:\Users\negar\anaconda3\Lib\subprocess.py", line 548, in run
    with Popen(*popenargs, **kwargs) as process:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\negar\anaconda3\Lib\subprocess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "C:\Users\negar\anaconda3\Lib\subprocess.py", line 1538, in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing wa
s 0.214247 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 274287
[LightGBM] [Info] Number of data points in the train set: 37130, number of used feat
ures: 4973
[LightGBM] [Info] Start training from score -2.613426
[LightGBM] [Info] Start training from score -2.914302
[LightGBM] [Info] Start training from score -1.236454
[LightGBM] [Info] Start training from score -1.176785
[LightGBM] [Info] Start training from score -3.787589
[LightGBM] [Info] Start training from score -2.989557
[LightGBM] [Info] Start training from score -1.605272
Model: LogisticRegression
Accuracy: 0.7469993087412807
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.71      0.76      1167
           1       0.90      0.61      0.72       863
           2       0.69      0.73      0.71      4621
           3       0.80      0.96      0.87      4905
           4       0.65      0.43      0.52       360
           5       0.75      0.34      0.47       801
           6       0.69      0.65      0.67      3196

    accuracy                           0.75     15913
   macro avg       0.76      0.63      0.67     15913
weighted avg       0.75      0.75      0.74     15913

--------------------------------------------------
Model: DecisionTreeClassifier
Accuracy: 0.6563815748130459
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.62      0.62      1167
           1       0.63      0.53      0.58       863
           2       0.60      0.60      0.60      4621
           3       0.83      0.87      0.85      4905
           4       0.48      0.53      0.50       360
           5       0.43      0.41      0.42       801
           6       0.55      0.53      0.54      3196

    accuracy                           0.66     15913
   macro avg       0.59      0.58      0.59     15913
weighted avg       0.65      0.66      0.65     15913

--------------------------------------------------
Model: RandomForestClassifier
Accuracy: 0.7120593225664551
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.59      0.70      1167
           1       0.96      0.47      0.63       863
           2       0.58      0.79      0.67      4621
           3       0.82      0.94      0.88      4905
           4       0.64      0.38      0.47       360
           5       0.92      0.25      0.39       801
           6       0.70      0.52      0.59      3196

    accuracy                           0.71     15913
   macro avg       0.78      0.56      0.62     15913
weighted avg       0.74      0.71      0.70     15913

--------------------------------------------------
Model: XGBClassifier
Accuracy: 0.7602589078112235
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.73      0.77      1167
           1       0.90      0.72      0.80       863
           2       0.70      0.74      0.72      4621
           3       0.83      0.94      0.88      4905
           4       0.68      0.61      0.64       360
           5       0.67      0.43      0.52       801
           6       0.70      0.64      0.67      3196

    accuracy                           0.76     15913
   macro avg       0.76      0.69      0.71     15913
weighted avg       0.76      0.76      0.76     15913

--------------------------------------------------
Model: LGBMClassifier
Accuracy: 0.7745239741092189
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.76      0.79      1167
           1       0.88      0.74      0.81       863
           2       0.71      0.74      0.73      4621
           3       0.87      0.94      0.90      4905
           4       0.70      0.66      0.68       360
           5       0.68      0.48      0.56       801
           6       0.69      0.67      0.68      3196

    accuracy                           0.77     15913
   macro avg       0.76      0.71      0.73     15913
weighted avg       0.77      0.77      0.77     15913

--------------------------------------------------
Model: MLPClassifier
Accuracy: 0.726010180355684
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.71      0.74      1167
           1       0.78      0.70      0.74       863
           2       0.68      0.68      0.68      4621
           3       0.87      0.89      0.88      4905
           4       0.61      0.68      0.64       360
           5       0.53      0.51      0.52       801
           6       0.61      0.62      0.62      3196

    accuracy                           0.73     15913
   macro avg       0.69      0.68      0.69     15913
weighted avg       0.73      0.73      0.73     15913

--------------------------------------------------
```