```
!pip install scikit-plot
```

```
Requirement already satisfied: scikit-plot in /usr/local/lib/python3.10/dist-packages (0.3.7)
Requirement already satisfied: matplotlib>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (3.7.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.3.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.13.1)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (0.12.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (4
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.26.4
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (24
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (9.4.0
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (3
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->scikit-plot
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=1.4.0->sc
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

import re
import string
from wordcloud import WordCloud
from collections import Counter
import warnings
warnings.filterwarnings('ignore')

from nltk import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
data = pd.read_excel ("/content/dataset.xlsx")
data.head()
```

	text	label
0	oh my gosh	1.0
1	trouble sleeping, confused mind, restless hear...	1.0
2	All wrong, back off dear, forward doubt. Stay ...	1.0
3	I've shifted my focus to something else but I'...	1.0
4	I'm restless and restless, it's been a month n...	1.0

```
print(data.shape)
```

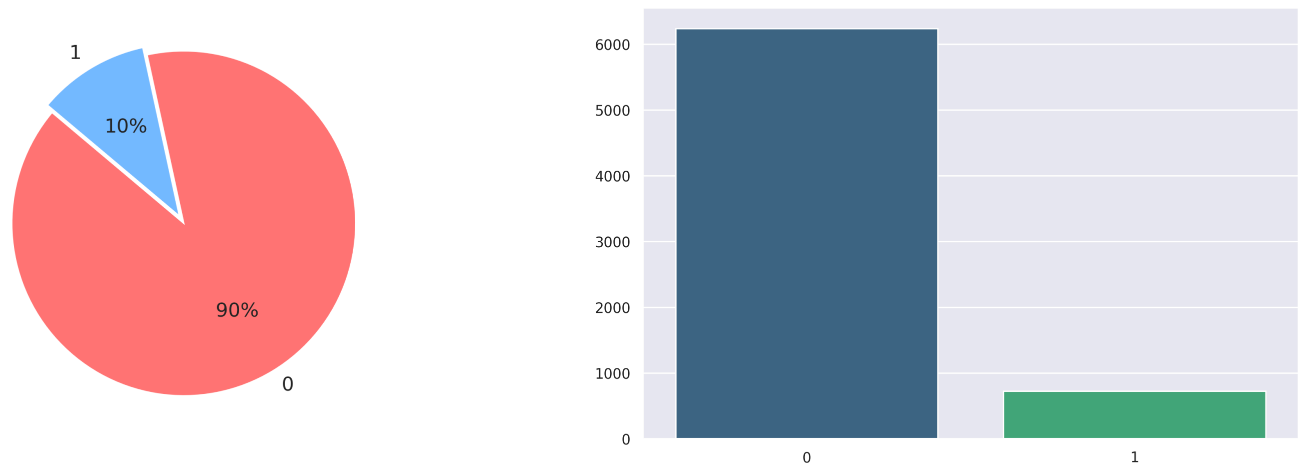
```
(6982, 2)
```

```
data=data.dropna(how='any')
```

```
data['label'].value_counts()
```

	count
label	
0.0	6240
1.0	730

```
labels = [0,1]
sizes = [6240, 730]
custom_colours = ['#ff7675', '#74b9ff']

plt.figure(figsize=(20, 6), dpi=227)
plt.subplot(1, 2, 1)
plt.pie(sizes, labels = labels, textprops={'fontsize': 15}, startangle=140,
       autopct='%1.0f%%', colors=custom_colours, explode=[0, 0.05])

plt.subplot(1, 2, 2)
sns.barplot(x=labels,y = sizes, palette= 'viridis')

plt.show()
```

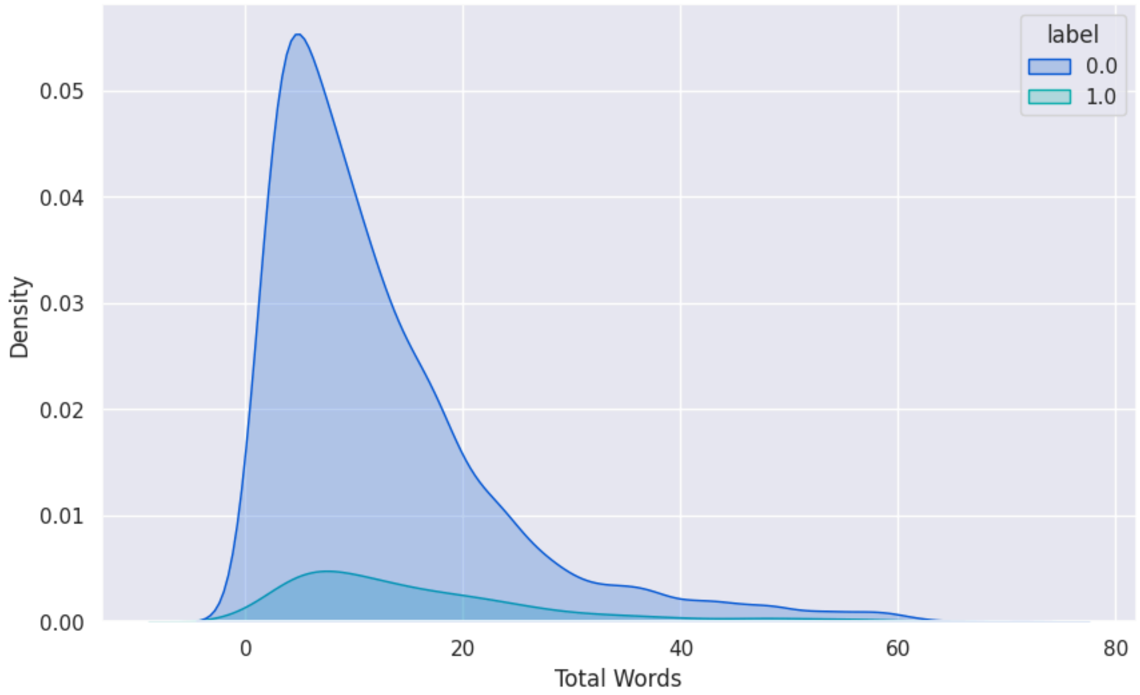


```
data['Total Words'] = data['text'].apply(lambda x: len(x.split()))

def count_total_words(text):
    char = 0
    for word in text.split():
        char += len(word)
    return char

data['Total Chars'] = data["text"].apply(count_total_words)
```

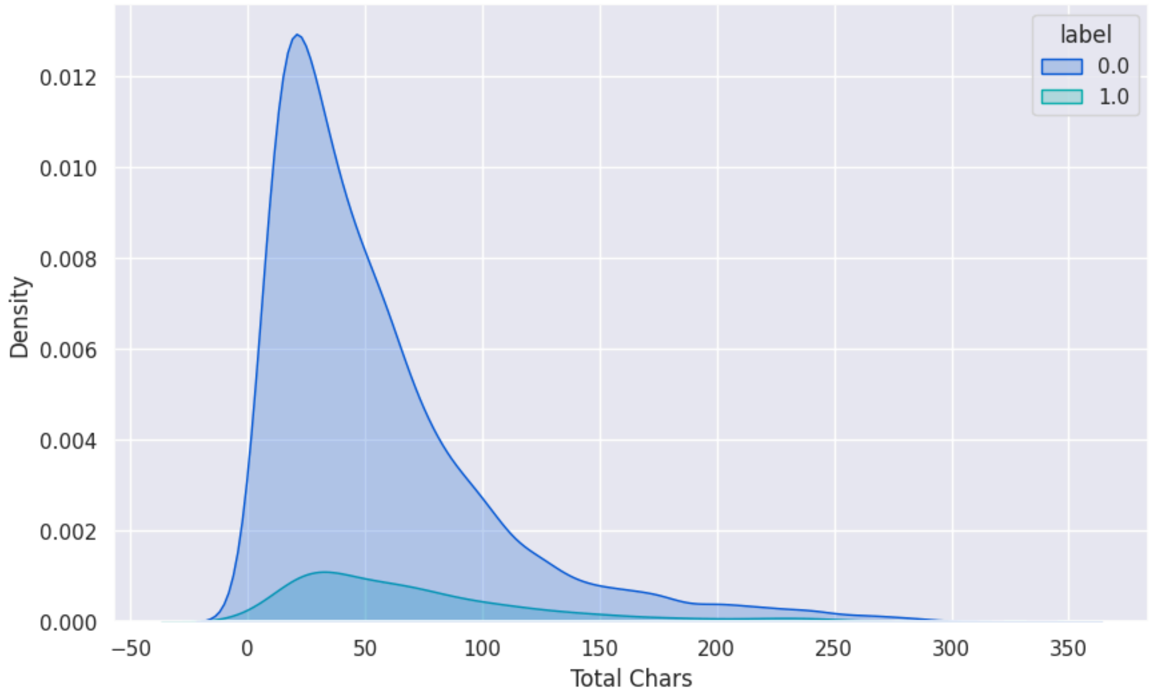
```
data.head()
```

	text	label	Total Words	Total Chars
0	oh my gosh	1.0	3	8
1	trouble sleeping, confused mind, restless hear...	1.0	10	55
2	All wrong, back off dear, forward doubt. Stay ...	1.0	14	65
3	I've shifted my focus to something else but I'...	1.0	11	51
4	I'm restless and restless, it's been a month n...	1.0	14	59

```
plt.figure(figsize = (10, 6))
sns.kdeplot(x = data['Total Words'], hue= data['label'], palette= 'winter', shade = True)
plt.show()
```



```
plt.figure(figsize = (10, 6))
sns.kdeplot(x = data['Total Chars'], hue= data['label'], palette= 'winter', shade = True)
plt.show()
```



```
data.head()
```

	text	label	Total Words	Total Chars
0	oh my gosh	1.0	3	8
1	trouble sleeping, confused mind, restless hear...	1.0	10	55
2	All wrong, back off dear, forward doubt. Stay ...	1.0	14	65
3	I've shifted my focus to something else but I'...	1.0	11	51
4	I'm restless and restless, it's been a month n...	1.0	14	59

```
def convert_lowercase(text):
    text = text.lower()
    return text

data['text'] = data['text'].apply(convert_lowercase)
```

```
def remove_url(text):
    re_url = re.compile('https?://\S+|www\.\S+')
    return re_url.sub('', text)

data['text'] = data['text'].apply(remove_url)
```

```
exclude = string.punctuation

def remove_punc(text):
    return text.translate(str.maketrans('', '', exclude))

data['text'] = data['text'].apply(remove_punc)
```

```
import nltk
nltk.download('punkt')
def remove_stopwords(text):
    new_list = []
    words = word_tokenize(text)
    stopwrds = stopwords.words('english')
    for word in words:
        if word not in stopwrds:
            new_list.append(word)
    return ' '.join(new_list)

data['text'] = data['text'].apply(remove_stopwords)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

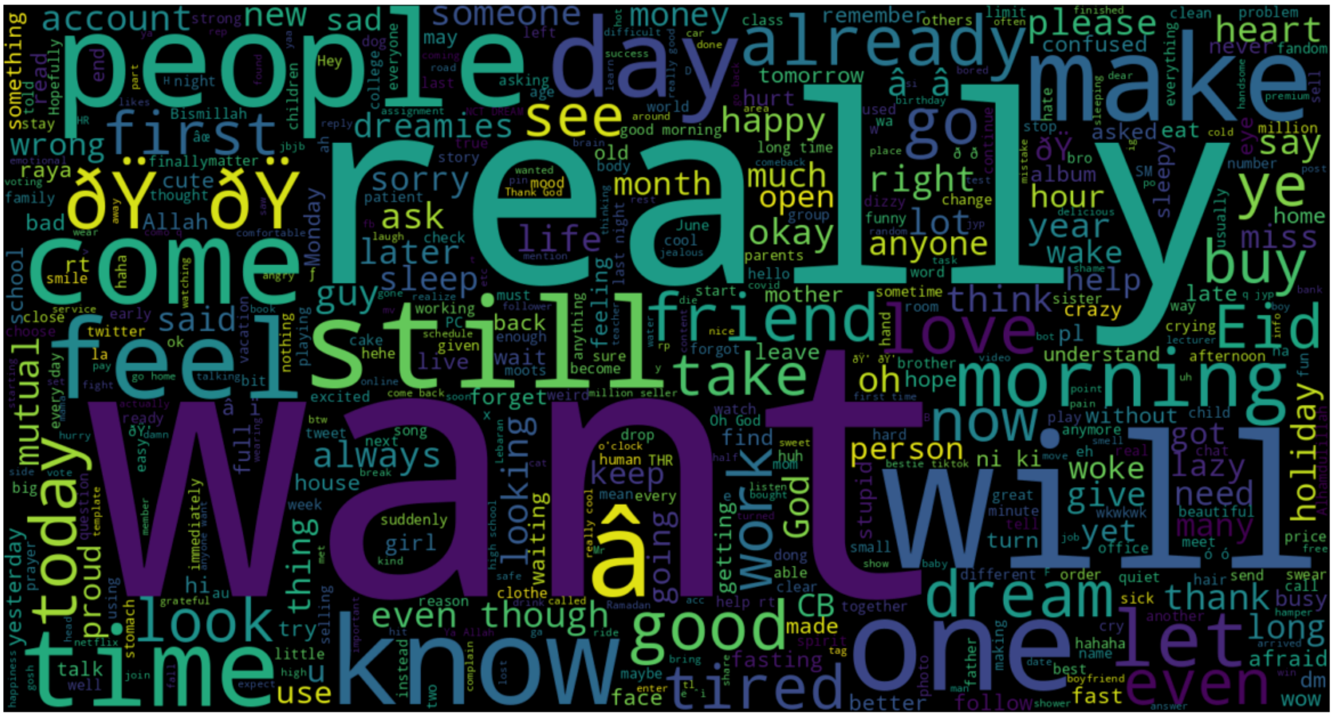
```
def perform_stemming(text):
    stemmer = PorterStemmer()
    new_list = []
    words = word_tokenize(text)
    for word in words:
        new_list.append(stemmer.stem(word))

    return " ".join(new_list)

data['text'] = data['text'].apply(perform_stemming)
```

```
data['Total Words After Transformation'] = data['text'].apply(lambda x: np.log(len(x.split())))
```

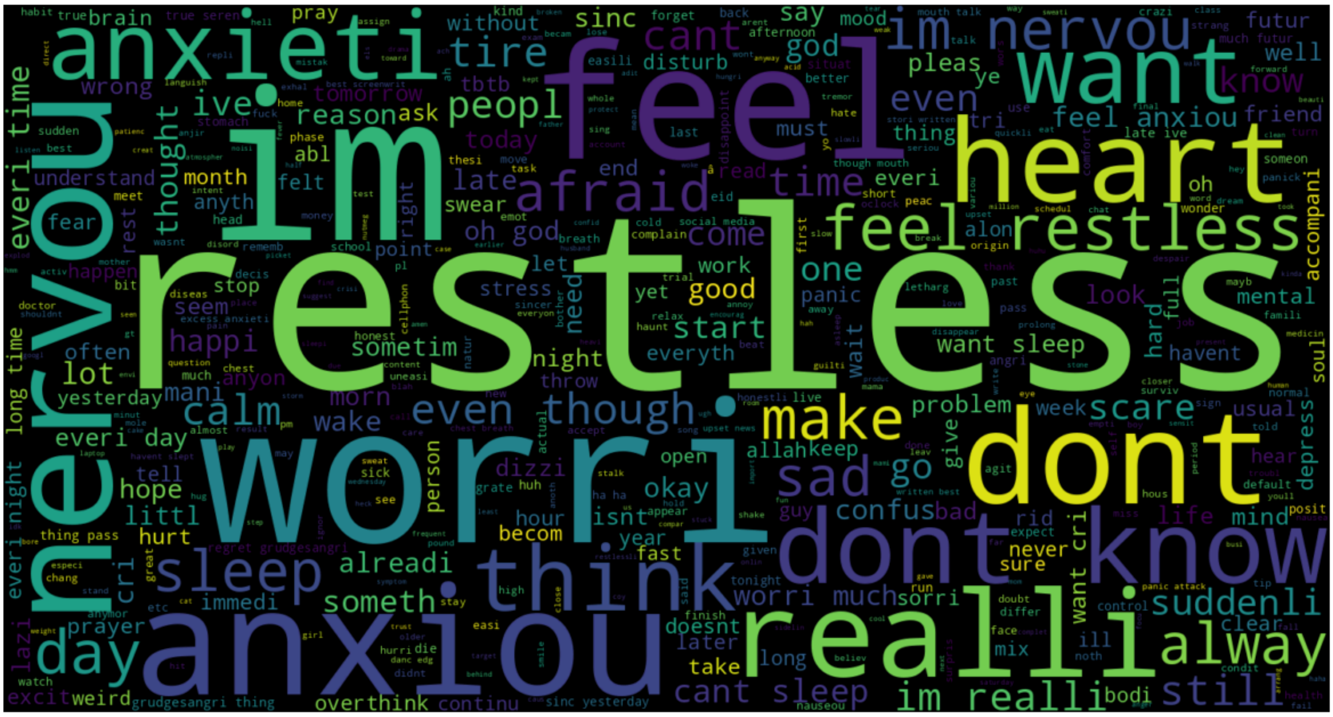
```
data.head()
```

	text	label	Total Words	Total Chars	Total Words After Transformation
0	oh gosh	1.0	3	8	0.693147
1	troubl sleep confus mind restless heart tune	1.0	10	55	1.945910
2	wrong back dear forward doubt stay restless re...	1.0	14	65	2.197225
3	ive shift focu someth els im still worri	1.0	11	51	2.079442
4	im restless restless month boy mean	1.0	14	59	1.791759

```
text = " ".join(data[data['label'] == 0]['text'])
plt.figure(figsize = (15, 10))
wordcloud = WordCloud(max_words=500, height= 800, width = 1500,  background_color="black", colormap= 'viridis').generate(text)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

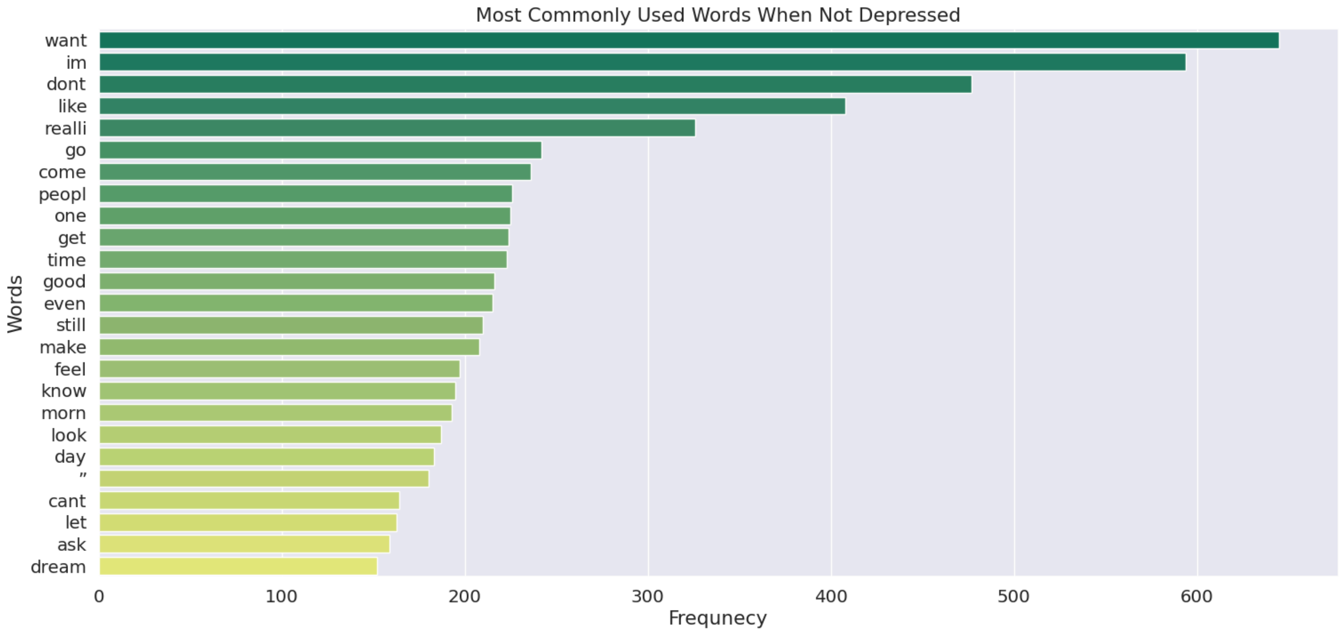


```
text = " ".join(data[data['label'] == 1]['text'])
plt.figure(figsize = (15, 10))
wordcloud = WordCloud(max_words=500, height= 800, width = 1500,  background_color="black", colormap= 'viridis').generate(text)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

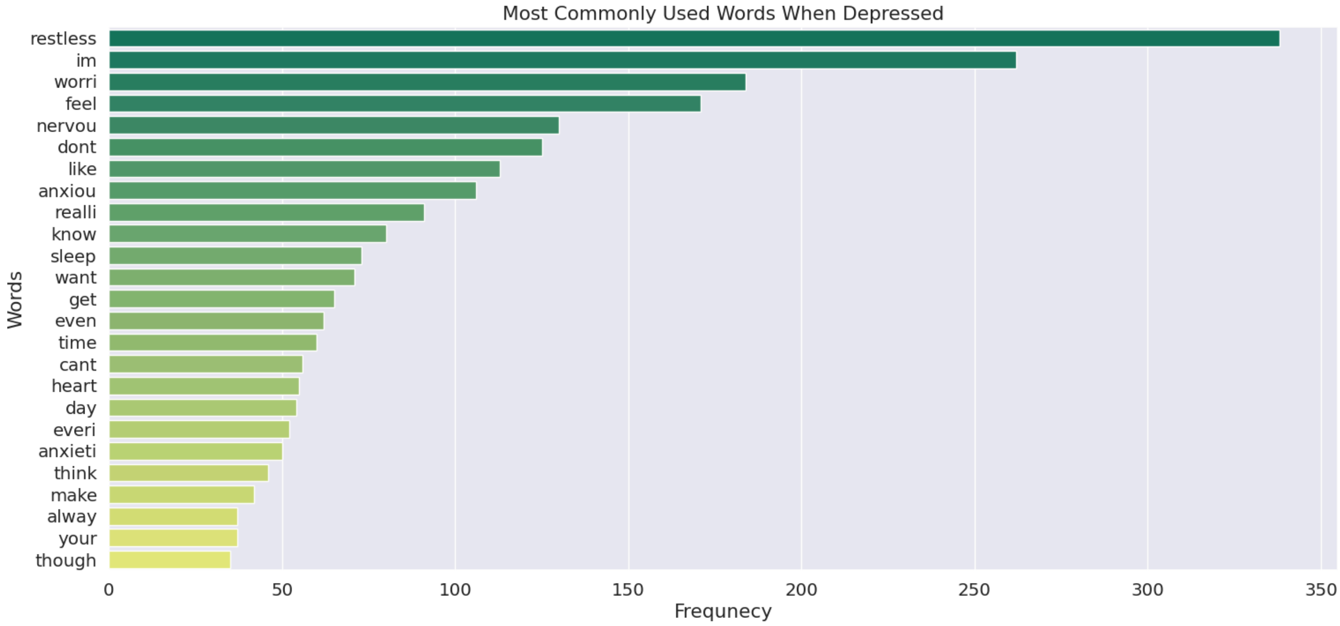


```
all_nodep_words = []
for sentence in data[data['label'] == 0]['text'].to_list():
    for word in sentence.split():
        all_nodep_words.append(word)

df = pd.DataFrame(Counter(all_nodep_words).most_common(25), columns= ['Word', 'Frequency'])
```

```
sns.set_context('notebook', font_scale= 1.3)
plt.figure(figsize=(18,8))
sns.barplot(y = df['Word'], x= df['Frequency'], palette= 'summer')
plt.title("Most Commonly Used Words When Not Depressed")
plt.xlabel("Frequnecy")
plt.ylabel("Words")
plt.show()
```



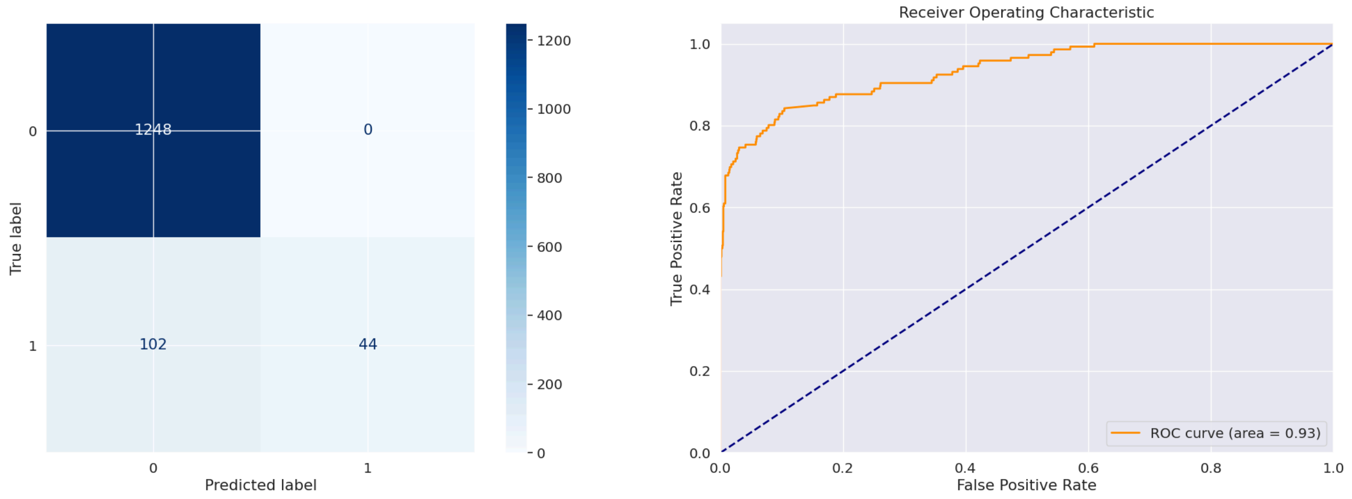
```
all_dep_words = []
for sentence in data[data['label'] == 1]['text'].to_list():
    for word in sentence.split():
        all_dep_words.append(word)

df = pd.DataFrame(Counter(all_dep_words).most_common(25), columns= ['Word', 'Frequency'])

sns.set_context('notebook', font_scale= 1.3)
plt.figure(figsize=(18,8))
sns.barplot(y = df['Word'], x= df['Frequency'], palette= 'summer')
plt.title("Most Commonly Used Words When Depressed")
plt.xlabel("Frequnecy")
plt.ylabel("Words")
plt.show()
```



```
X = data["text"]
y = data['label'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 42, stratify = y)


tfidf = TfidfVectorizer(max_features= 2500, min_df= 2)
X_train = tfidf.fit_transform(X_train).toarray()
X_test = tfidf.transform(X_test).toarray()
```

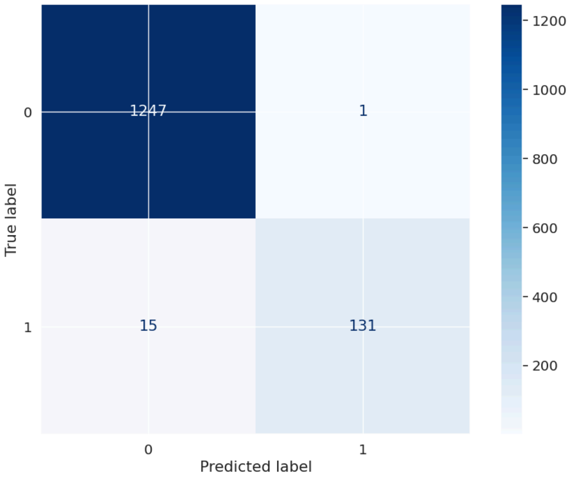
```
def train_model(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)

    print(f'Accuracy of the model: {accuracy}')
    print(f'Precision Score of the model: {precision}')
    print(f'Recall Score of the model: {recall}')

    sns.set_context('notebook', font_scale= 1.3)
    fig, ax = plt.subplots(1, 2, figsize = (25,  8))

    # Import necessary function for confusion matrix plotting
    from sklearn.metrics import ConfusionMatrixDisplay

    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap=plt.cm.Blues, ax=ax[0])  # Plot on the first subplot

    from sklearn.metrics import roc_curve, roc_auc_score

    # Assuming you have y_true (true labels) and y_score (predicted probabilities)
    fpr, tpr, _ = roc_curve(y_test, y_prob[:, 1]) # Use y_prob for class 1
    roc_auc = roc_auc_score(y_test, y_prob[:, 1])

    # Plot ROC on the second subplot
    ax[1].plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
    ax[1].plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    ax[1].set_xlim([0.0, 1.0])
    ax[1].set_ylim([0.0, 1.05])
    ax[1].set_xlabel('False Positive Rate')
    ax[1].set_ylabel('True Positive Rate')
    ax[1].set_title('Receiver Operating Characteristic')
    ax[1].legend(loc="lower right")

    plt.show() # Dedent plt.show() to be outside the function

nb = MultinomialNB()
train_model(nb)
```

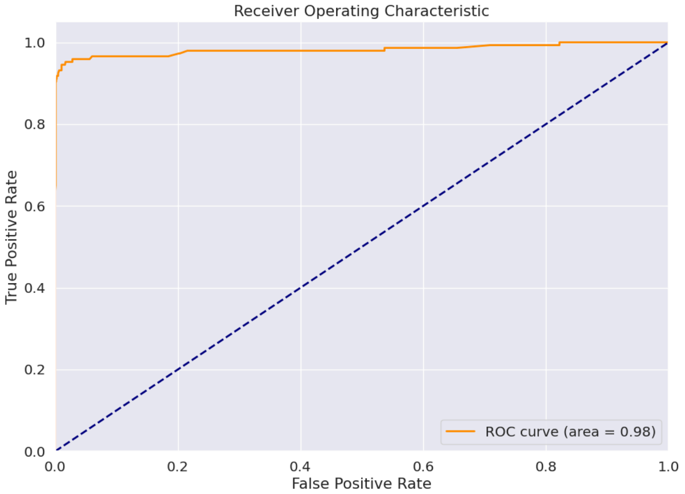
```
Accuracy of the model: 0.927
Precision Score of the model: 1.0
Recall Score of the model: 0.301
```



```
rf = RandomForestClassifier(n_estimators= 300)
train_model(rf)
```

```
Accuracy of the model: 0.989
Precision Score of the model: 0.992
Recall Score of the model: 0.897
```

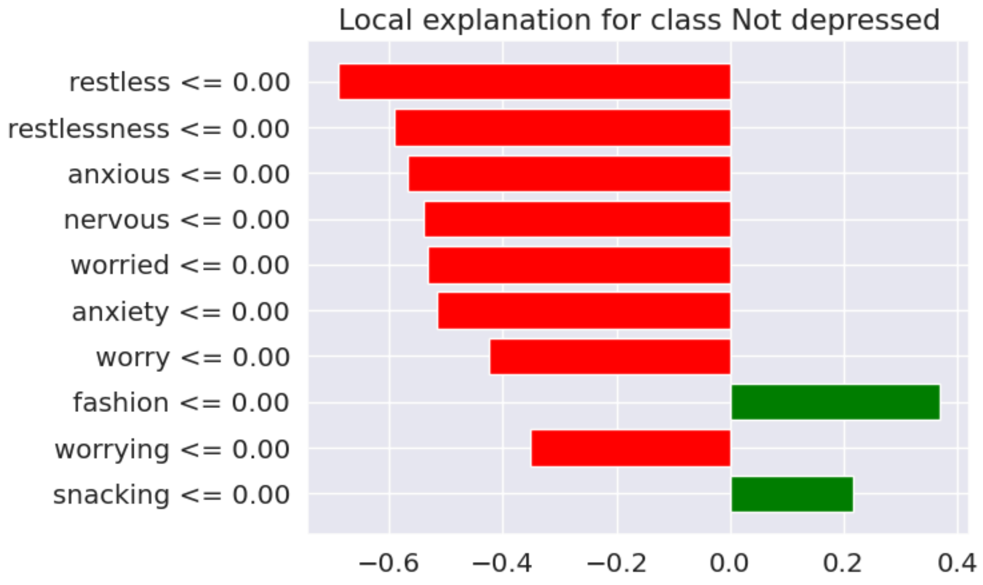




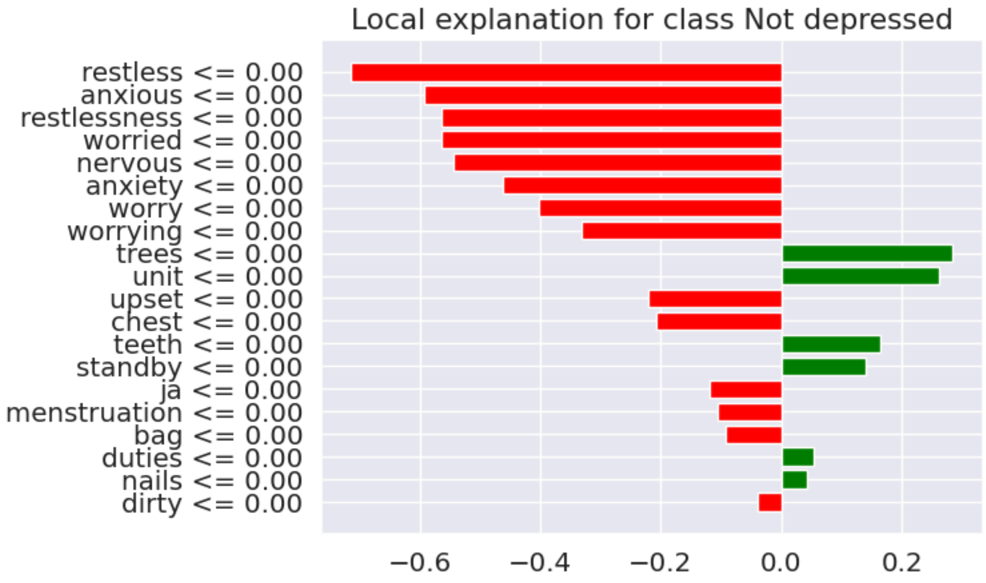
```
!pip install lime
# Import the LimeTabularExplainer module
from lime.lime_tabular import LimeTabularExplainer

# Get the class names
class_names = ['depressed', 'Not depressed']

# Get the feature names
feature_names = tfidf.get_feature_names_out()

# X_train is already a dense array, no need to call toarray()
X_train_dense = X_train

# Fit the Explainer on the training data set using the LimeTabularExplainer
explainer = LimeTabularExplainer(X_train_dense,
                                 feature_names=feature_names,
                                 class_names=class_names,
                                 mode='classification')
```

```
Requirement already satisfied: lime in /usr/local/lib/python3.10/dist-packages (0.2.0.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from lime) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lime) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lime) (1.13.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lime) (4.66.5)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-packages (from lime) (1.3.2)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.10/dist-packages (from lime) (0.23.2)
Requirement already satisfied: networkx>=2.8 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (3.3)
Requirement already satisfied: pillow>=9.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (9.4.0)
Requirement already satisfied: imageio>=2.33 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (2.34.2)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (2024
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (24.1)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.12->lime) (0.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->lime) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->lime) (3.5
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->lime) (1
```

```
idx = 0  # For example, the first instance in your test set
instance = X_test[idx]

# Generate an explanation
explanation = explainer.explain_instance(instance, rf.predict_proba, num_features=10)

# Now you can plot the explanation
fig = explanation.as_pyplot_figure()
plt.show()
```

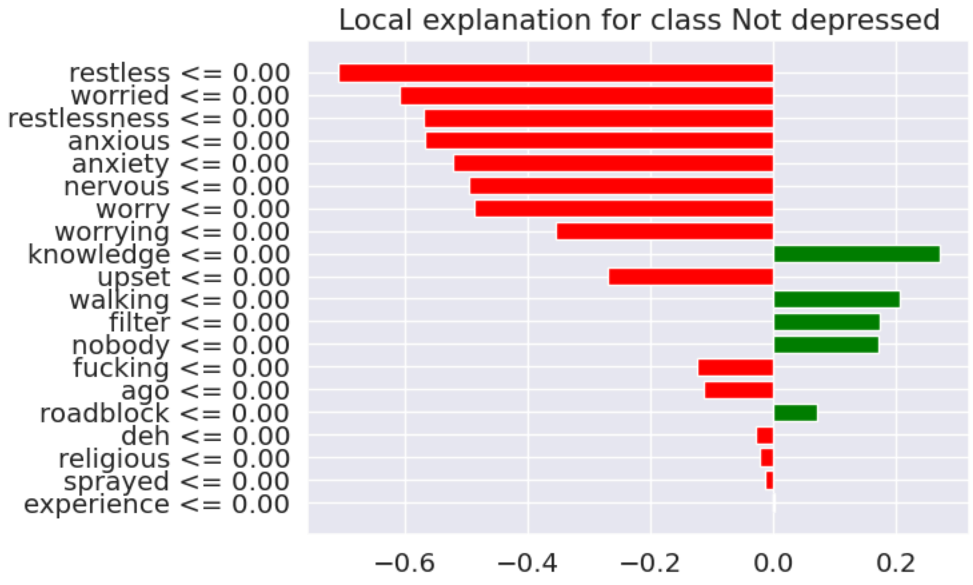


```
idx = 0  # For example, the first instance in your test set
instance = X_test[idx]

# Generate an explanation
explanation = explainer.explain_instance(instance, rf.predict_proba, num_features=20)

# Now you can plot the explanation
fig = explanation.as_pyplot_figure()
plt.show()
```



Start coding or generate with AI.

```
idx = 0  # For example, the first instance in your test set
instance = X_test[idx]

# Generate an explanation
explanation = explainer.explain_instance(instance, rf.predict_proba, num_features=20)

# Now you can plot the explanation
fig = explanation.as_pyplot_figure()
plt.show()
```



Local explanation for class Not depressed

Start coding or generate with AI.

```
!pip install lime
import lime.lime_text  # Import the correct module for text data
import numpy as np

# Assuming 'clf' is your trained text classifier (e.g., text_clf from previous examples)
explainer = lime.lime_text.LimeTextExplainer(
    class_names=['not depressed', 'depressed']  # Replace with your actual class names
)
idx = 2  # Index of the instance to explain

# **Make sure 'vectorizer' is defined here,
# it should be the same one used during training**
vectorizer = tfidf

exp = explainer.explain_instance(X.iloc[idx],
                                 lambda texts: rf.predict_proba(vectorizer.transform(texts)), # Vectorize the text before prediction
                                 num_features=5)

# Visualize the explanation
exp.as_pyplot_figure()
```