SENTIMENT ANALYSIS: A MACHINE LEARNING APPROACH

Analyzing Text Sentiment with VADER and NLP

- Introduction to Sentiment Analysis
- Natural Language Processing (NLP) Techniques
- Data Preprocessing
- Sentiment Analysis with VADER
- Visualizing Sentiment
- Model Saving
- Model loading

AGENDA

- **Definition:** Sentiment analysis, also known as opinion mining, is the process of determining the emotional tone behind a body of text. It is used to understand the attitudes, opinions, and emotions expressed within an online mention.

- **Applications:** Widely used in business to analyze customer feedback, social media monitoring, market research, and more.

- **Importance:** Helps businesses understand the customer sentiment towards products, services, or topics and make data-driven decisions.

WHAT IS SENTIMENT ANALYSIS?

NLP TECHNIQUES USED

- **Text Tokenization:** Breaking down text into individual words or sentences.

- **Stop Words Removal:** Eliminating common words that do not contribute to the sentiment (e.g., 'and', 'the').

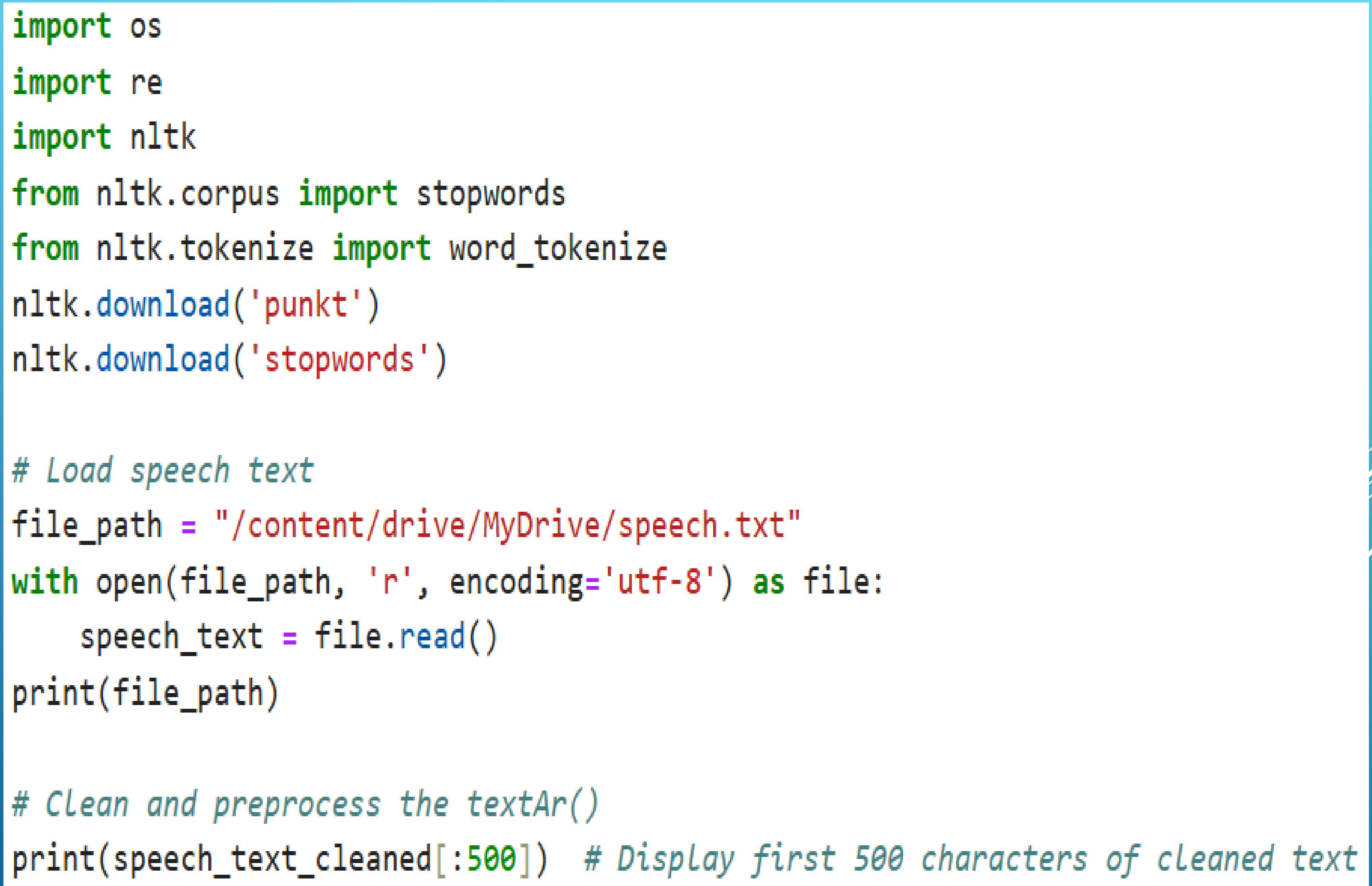
- **Lemmatization:** Reducing words to their base or root form.

- **Sentiment Analysis with VADER:** Using the VADER (Valence Aware Dictionary and sEntiment Reasoner) tool for analyzing the sentiment of text, particularly effective for social media content.

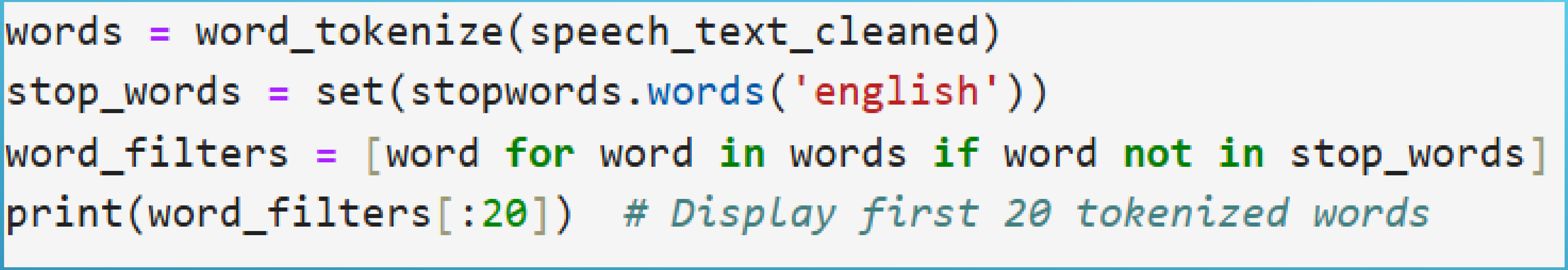
- **Loading Text Data:** Reading the speech text from a file.
- **Cleaning Text:** Removing punctuation and converting text to lowercase.
- **Tokenizing Text:** Splitting the text into individual words.
- **Removing Stop Words:** Filtering out common, non-informative words.
- **Lemmatizing Words:** Converting words to their base form for consistency.

DATA PREPROCESSING STEPS

LOADING AND CLEANING TEXT DATA

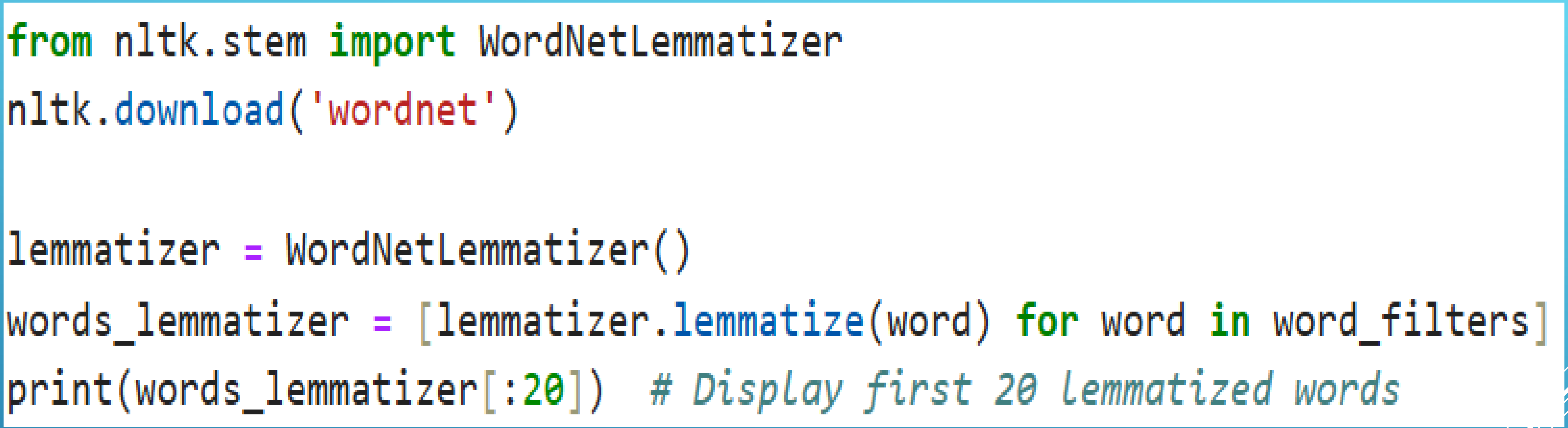


```
import os
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('stopwords')

# Load speech text
file_path = "/content/drive/MyDrive/speech.txt"
with open(file_path, 'r', encoding='utf-8') as file:
    speech_text = file.read()
print(file_path)

# Clean and preprocess the textAr()
print(speech_text_cleaned[:500])  # Display first 500 characters of cleaned text
```



```
words = word_tokenize(speech_text_cleaned)
stop_words = set(stopwords.words('english'))
word_filters = [word for word in words if word not in stop_words]
print(word_filters[:20])  # Display first 20 tokenized words
```

TOKENIZING AND REMOVING STOP WORDS



```python
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
words_lemmatizer = [lemmatizer.lemmatize(word) for word in word_filters]
print(words_lemmatizer[:20])  # Display first 20 lemmatized words
```

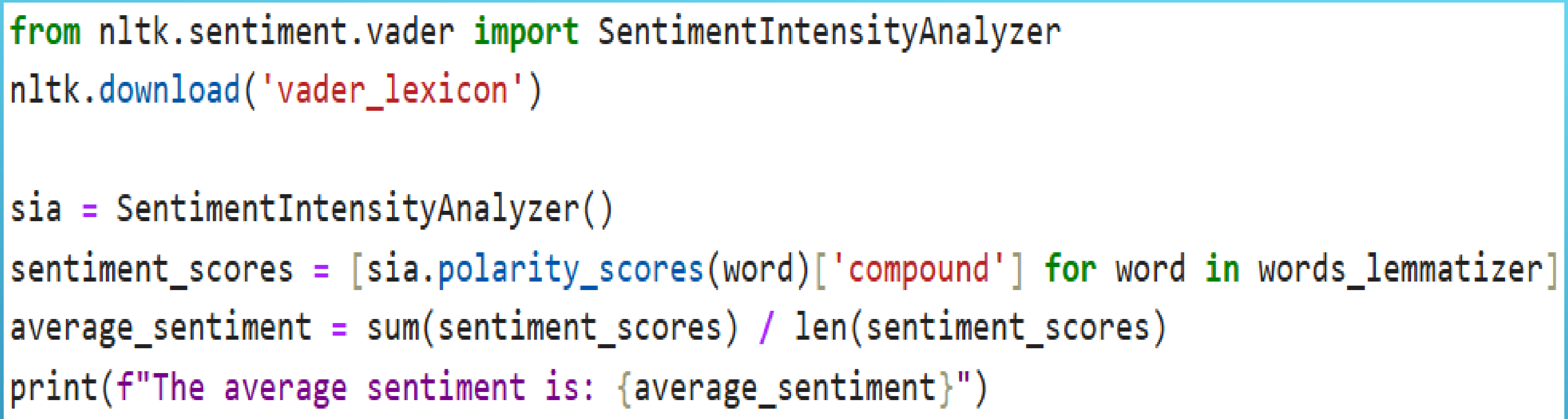
LEMMATIZATION

- **VADER Overview:** VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool designed for social media text.

- **Sentiment Scores:** VADER provides a compound sentiment score for each word, which is used to determine the overall sentiment of the text.

- **Categories:** Words are classified into positive, negative, and neutral categories based on their sentiment scores.

SENTIMENT ANALYSIS WITH VADER



```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')

sia = SentimentIntensityAnalyzer()
sentiment_scores = [sia.polarity_scores(word)['compound'] for word in words_lemmatizer]
average_sentiment = sum(sentiment_scores) / len(sentiment_scores)
print(f"The average sentiment is: {average_sentiment}")
```

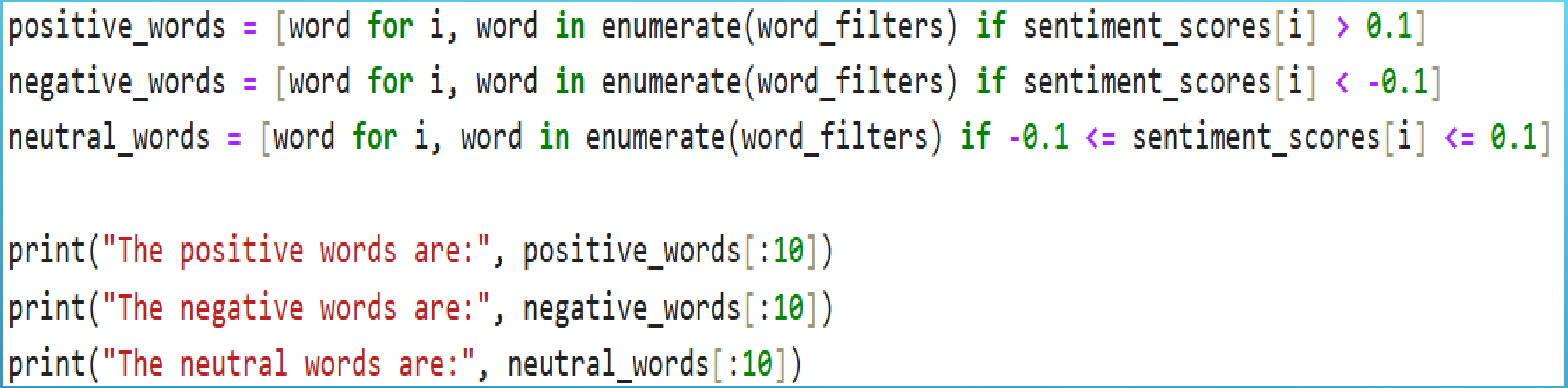
SENTIMENT ANALYSIS WITH VADER

VISUALIZING SENTIMENT

- **Word Clouds:** Visual representations of the most frequent positive and negative words.

- **Bar Charts:** Highlighting the top 10 most frequent words in each sentiment category.

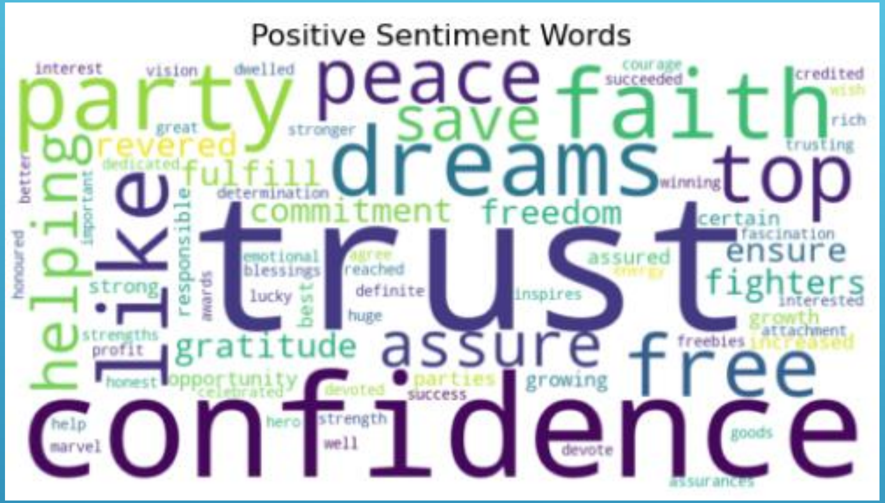
- **Pie Chart:** Showing the distribution of sentiments (positive, negative, neutral) across the text.



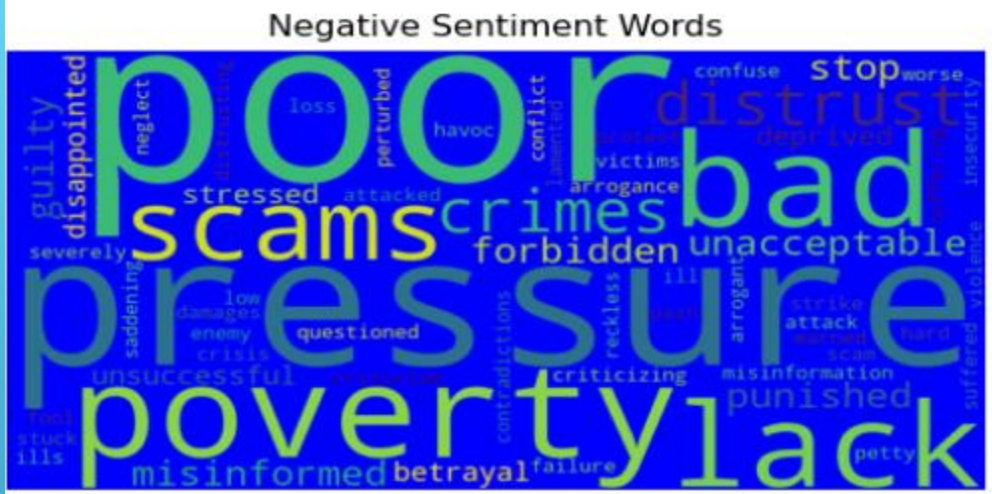
```
positive_words = [word for i, word in enumerate(word_filters) if sentiment_scores[i] > 0.1]
negative_words = [word for i, word in enumerate(word_filters) if sentiment_scores[i] < -0.1]
neutral_words = [word for i, word in enumerate(word_filters) if -0.1 <= sentiment_scores[i] <= 0.1]

print("The positive words are:", positive_words[:10])
print("The negative words are:", negative_words[:10])
print("The neutral words are:", neutral_words[:10])
```

CLASSIFYING WORDS BY SENTIMENT

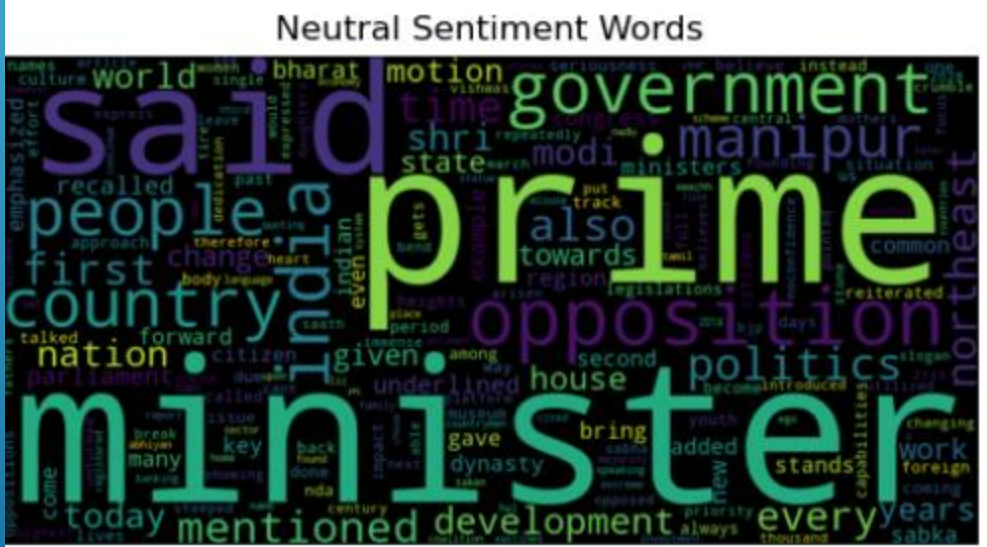
WORD CLOUD



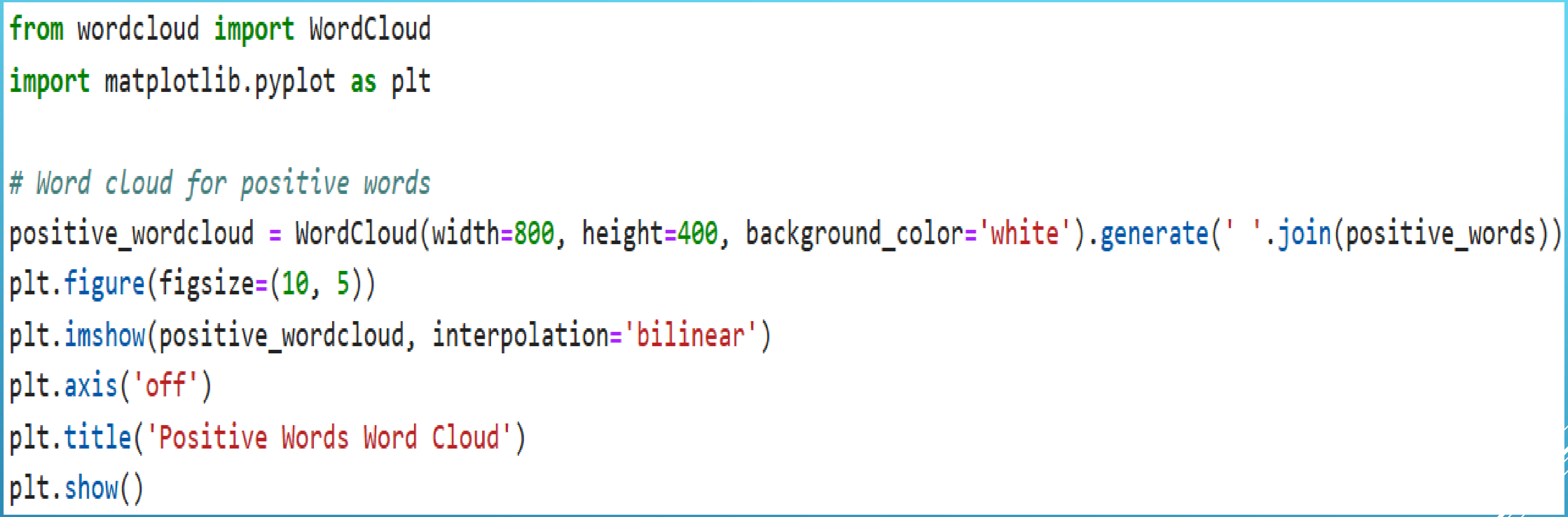
Positive Sentiment Words



Negative Sentiment Words

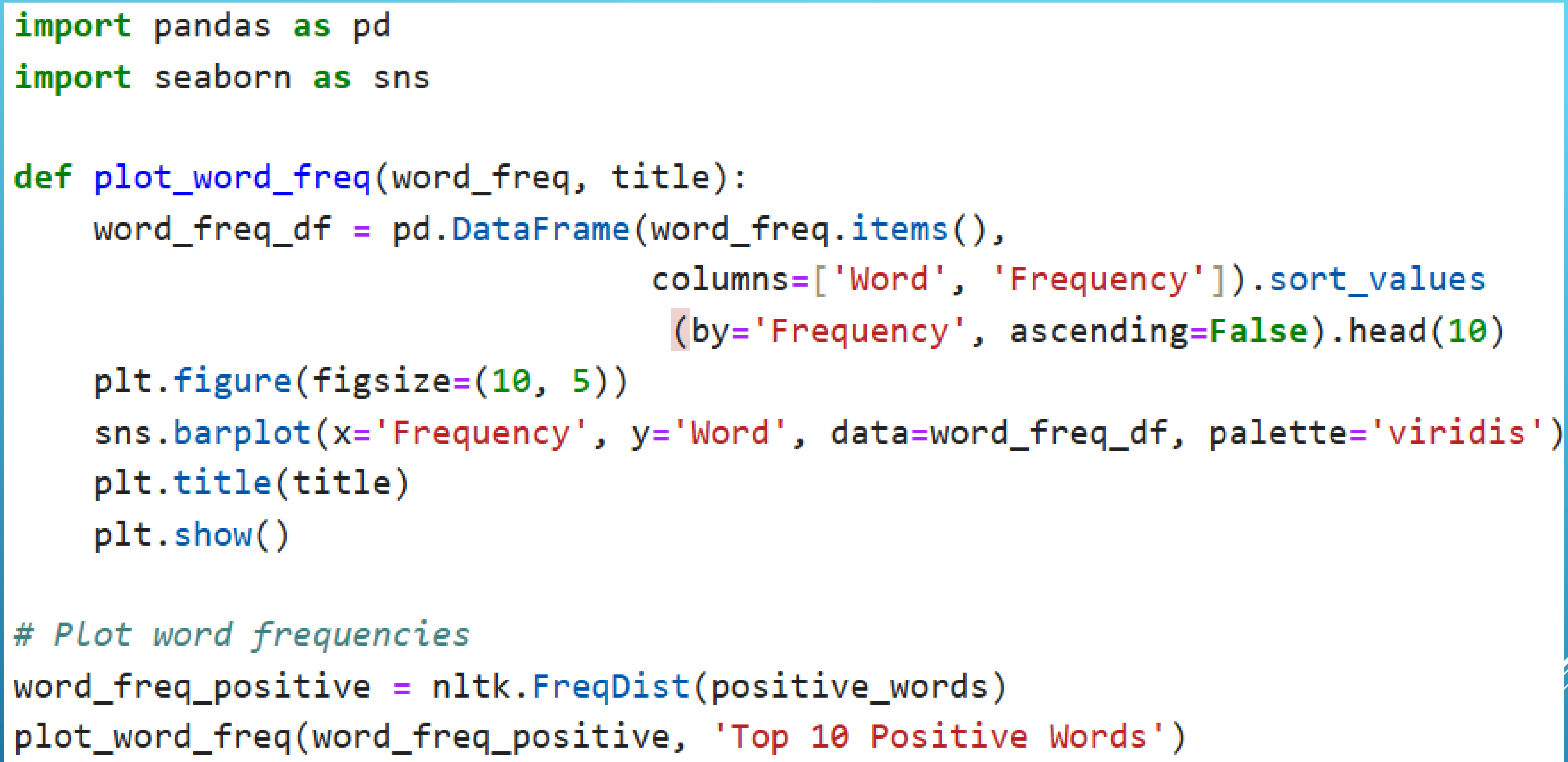


Neutral Sentiment Words



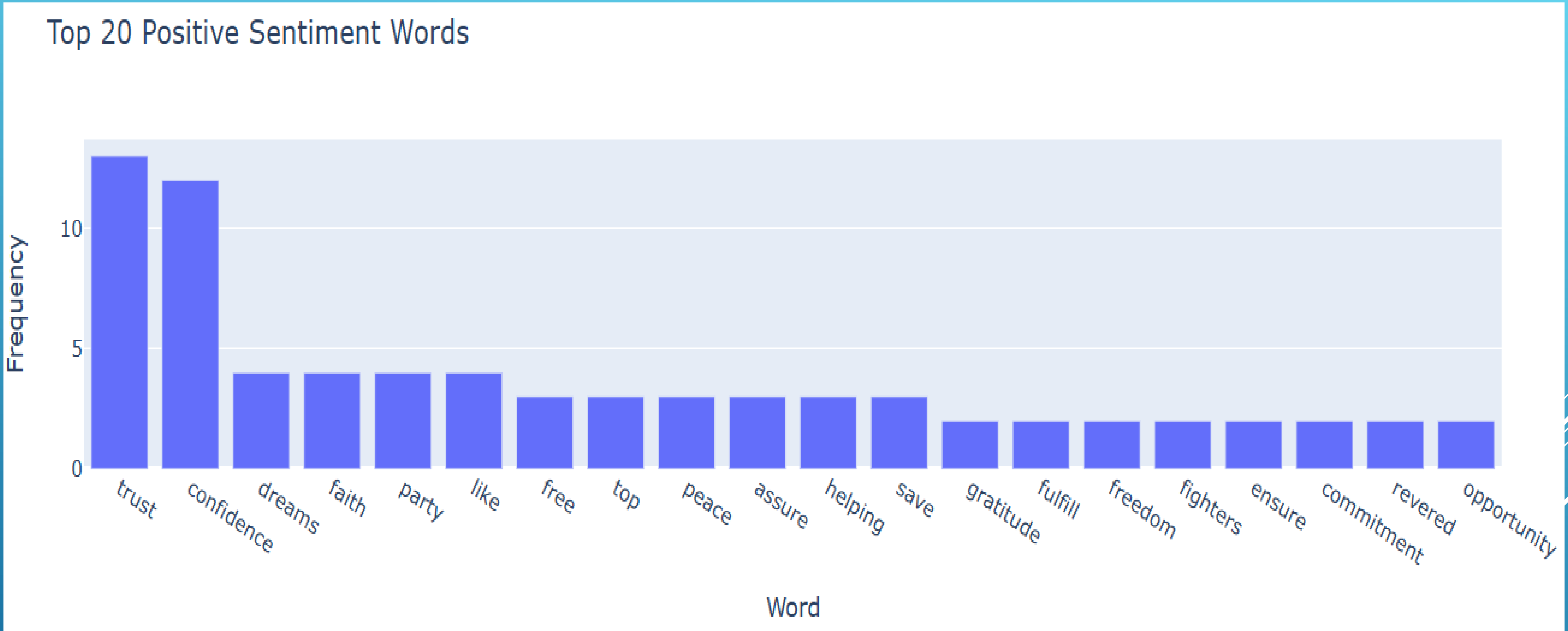
```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Word cloud for positive words
positive_wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' '.join(positive_words))
plt.figure(figsize=(10, 5))
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Positive Words Word Cloud')
plt.show()
```

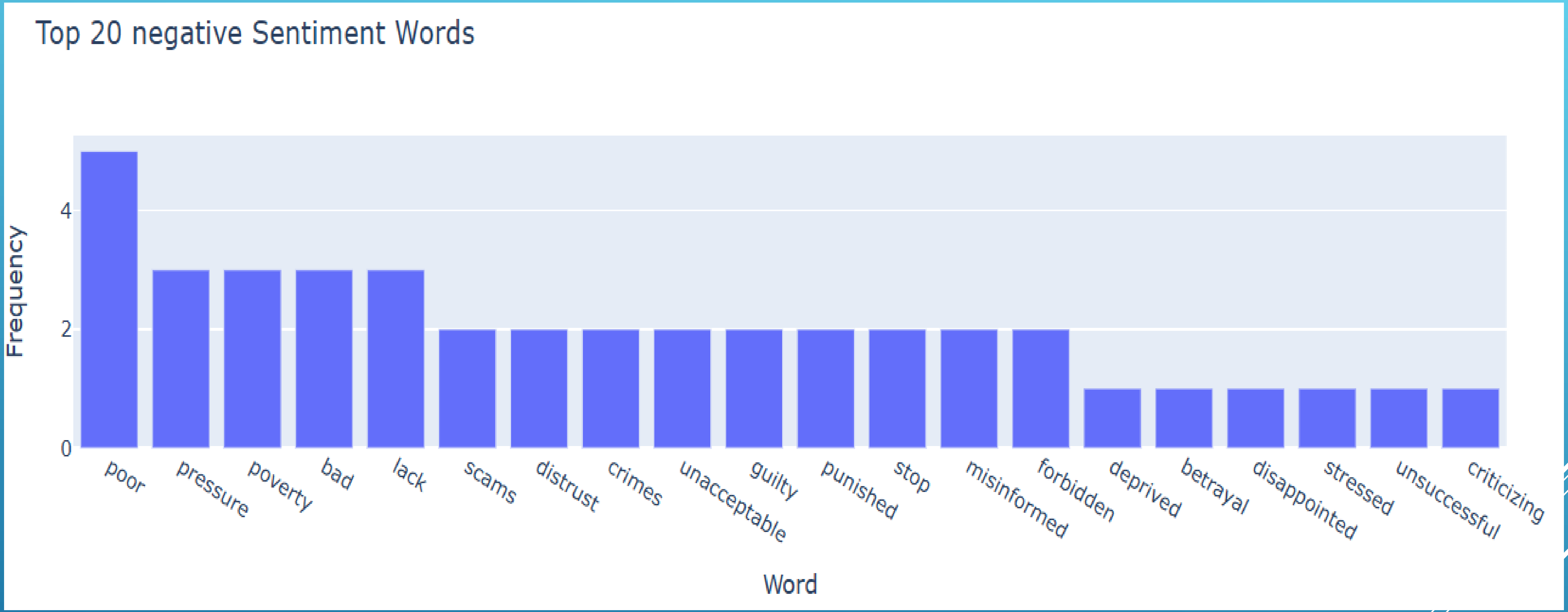
WORD CLOUDS FOR SENTIMENT

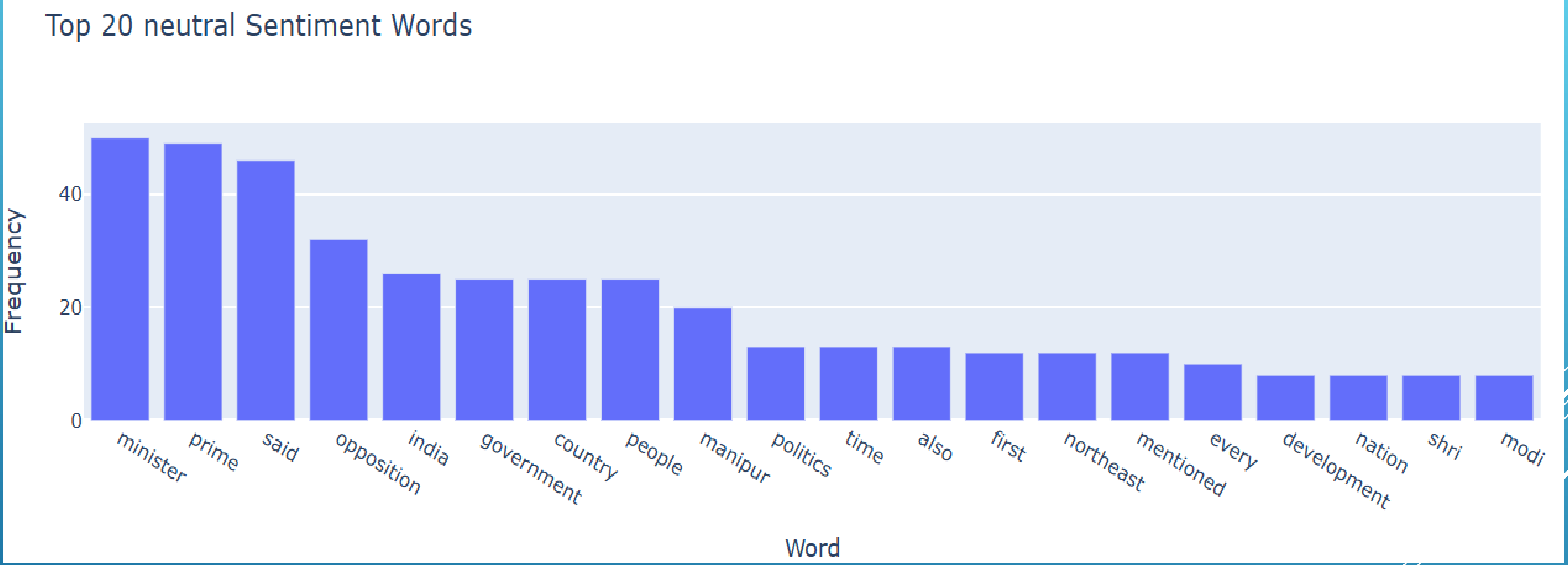


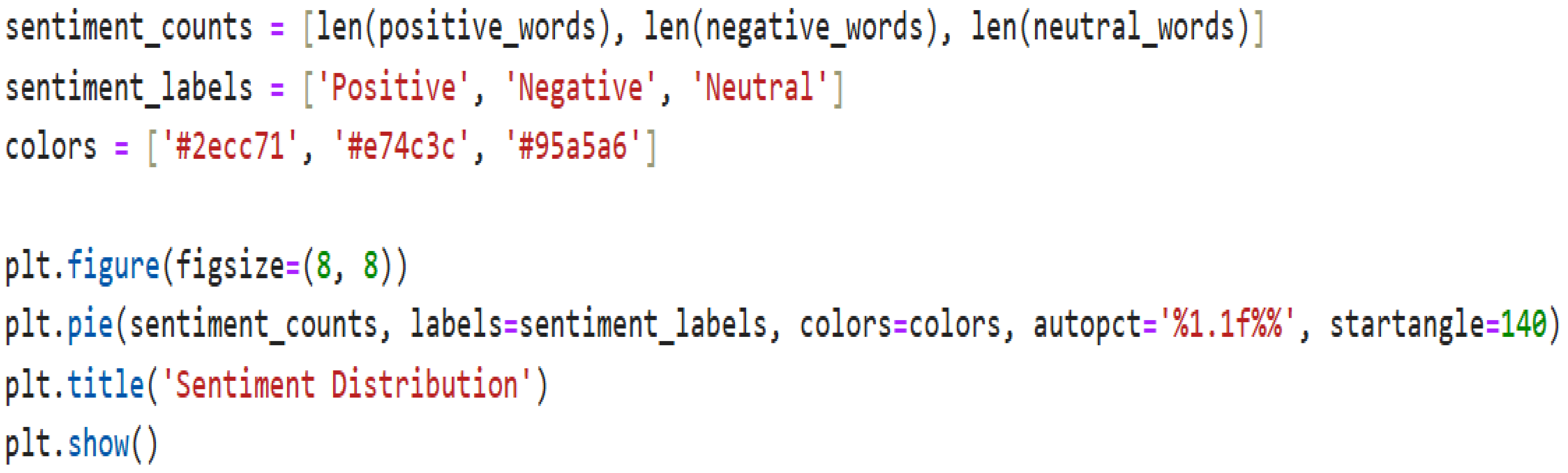
```
import pandas as pd
import seaborn as sns

def plot_word_freq(word_freq, title):
    word_freq_df = pd.DataFrame(word_freq.items(),
                                columns=['Word', 'Frequency']).sort_values
                                 (by='Frequency', ascending=False).head(10)
    plt.figure(figsize=(10, 5))
    sns.barplot(x='Frequency', y='Word', data=word_freq_df, palette='viridis')
    plt.title(title)
    plt.show()

# Plot word frequencies
word_freq_positive = nltk.FreqDist(positive_words)
plot_word_freq(word_freq_positive, 'Top 10 Positive Words')
```

BAR CHARTS FOR WORD FREQUENCIES


Top 20 Positive Sentiment Words
Frequency
0
5
10
trust
confidence
dreams
faith
party
like
free
top
peace
assure
helping
save
gratitude
fulfill
freedom
fighters
ensure
commitment
revered
opportunity
Word

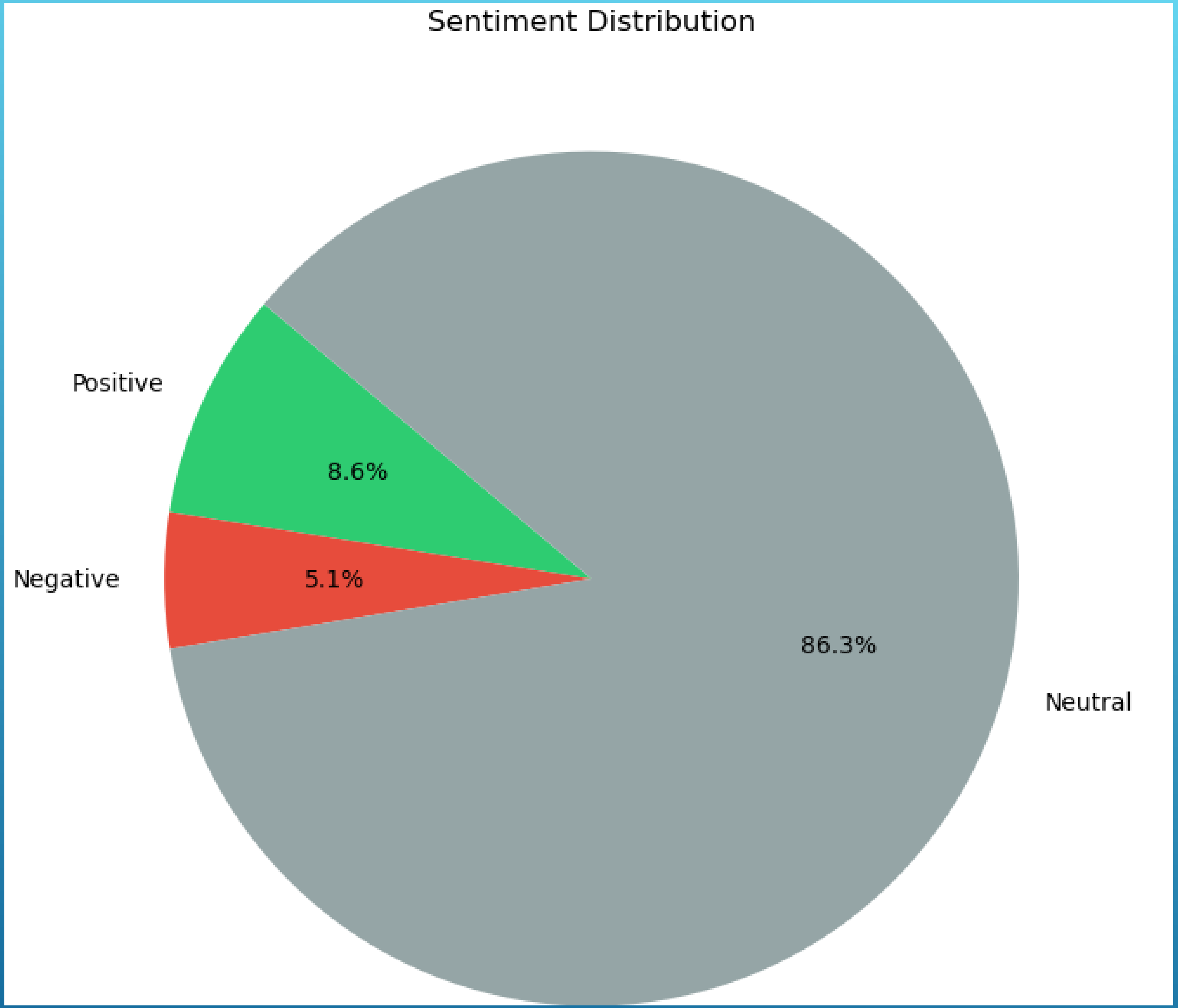

Top 20 negative Sentiment Words
Frequency
0
2
4
poor
pressure
poverty
bad
lack
scams
distrust
crimes
unacceptable
guilty
punished
stop
misinformed
forbidden
deprived
betrayal
disappointed
stressed
unsuccessful
criticizing
Word


Top 20 neutral Sentiment Words
Frequency
0
20
40
minister
prime
said
opposition
india
government
country
people
manipur
politics
time
also
first
northeast
mentioned
every
development
nation
shri
modi
Word

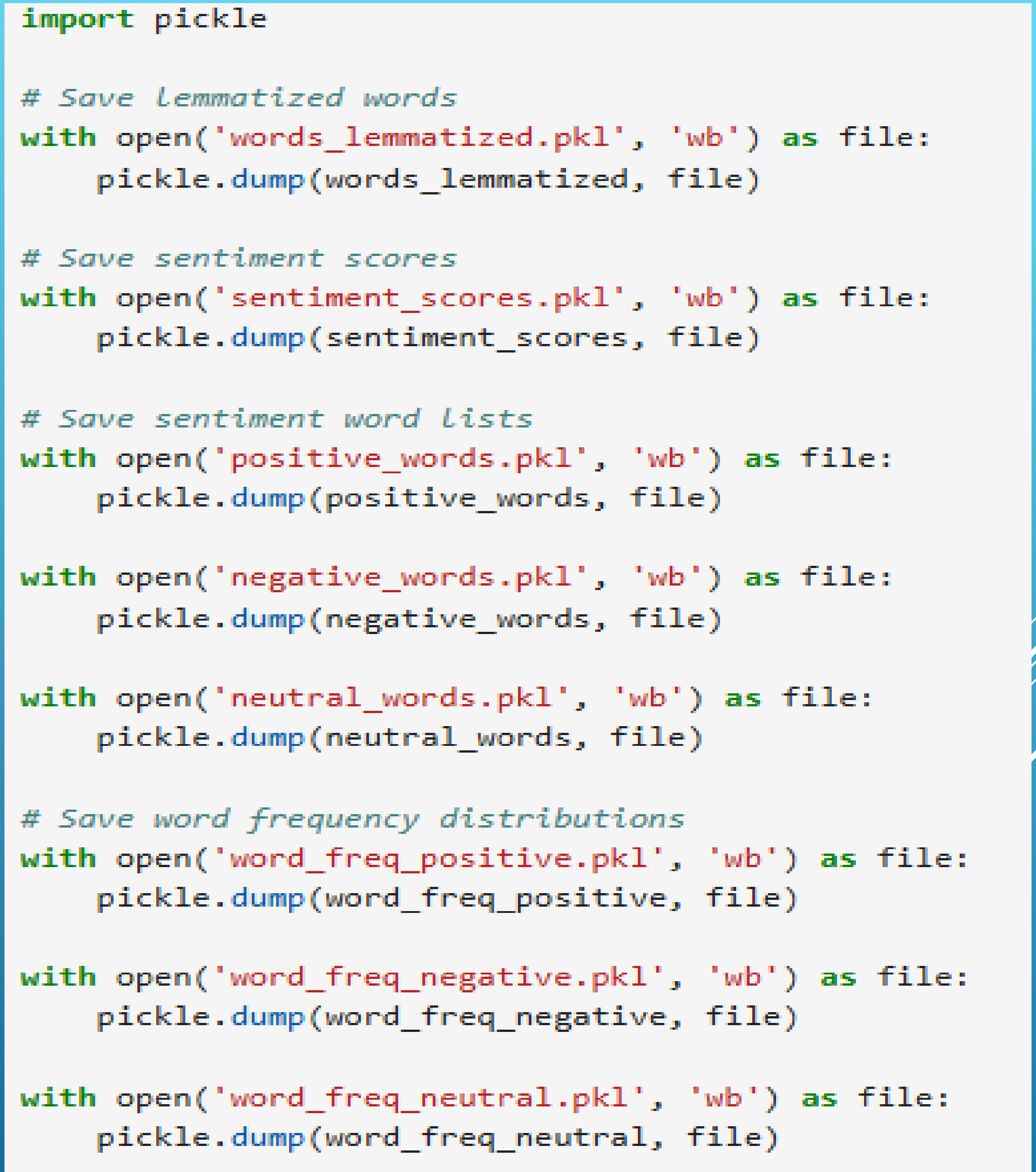


```
sentiment_counts = [len(positive_words), len(negative_words), len(neutral_words)]
sentiment_labels = ['Positive', 'Negative', 'Neutral']
colors = ['#2ecc71', '#e74c3c', '#95a5a6']

plt.figure(figsize=(8, 8))
plt.pie(sentiment_counts, labels=sentiment_labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Sentiment Distribution')
plt.show()
```

SENTIMENT DISTRIBUTION PIE CHART

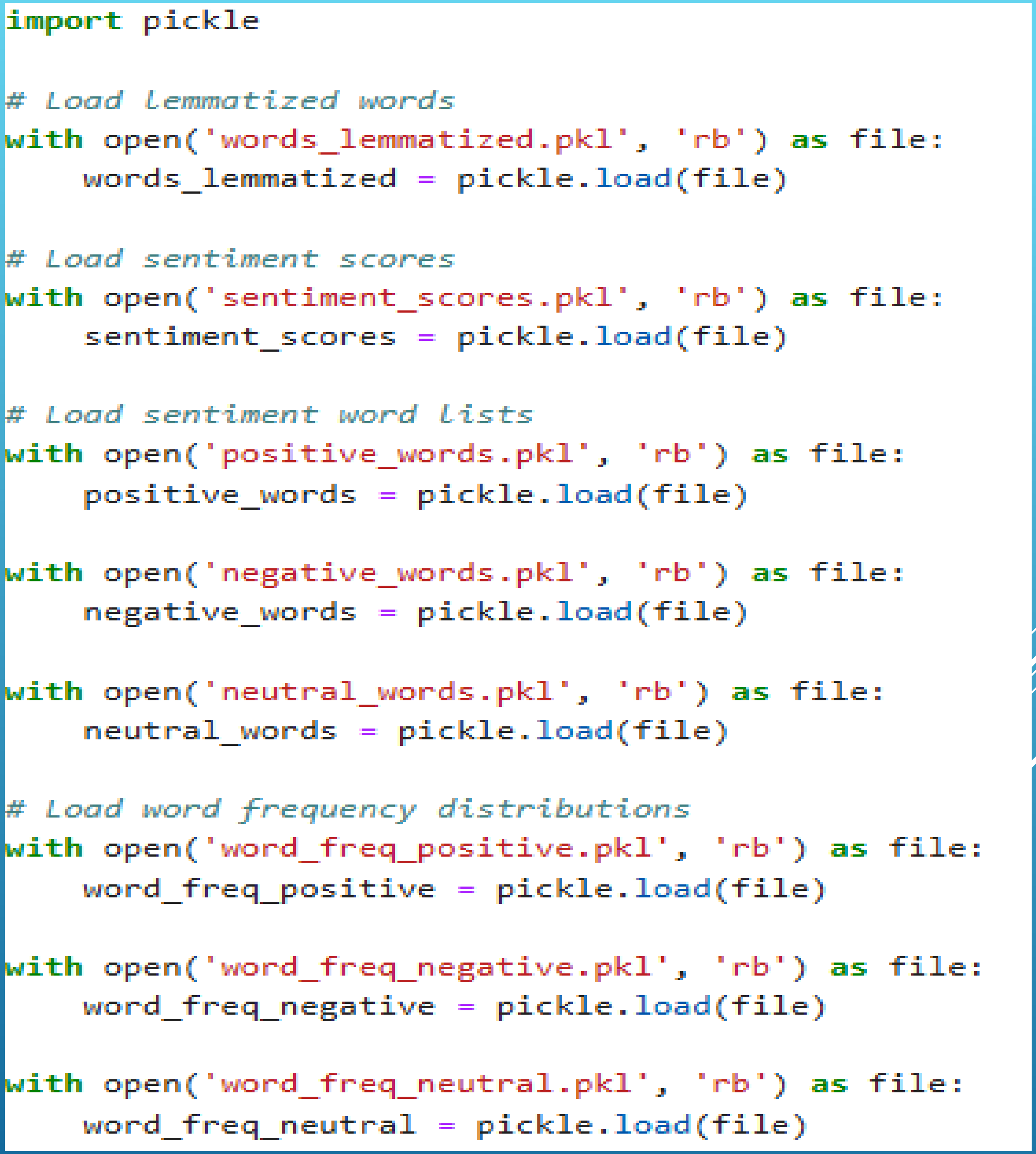

Sentiment Distribution
Positive
8.6%
Negative
5.1%
86.3%
Neutral

MODEL SAVING



```
import pickle

# Save lemmatized words
with open('words_lemmatized.pkl', 'wb') as file:
    pickle.dump(words_lemmatized, file)

# Save sentiment scores
with open('sentiment_scores.pkl', 'wb') as file:
    pickle.dump(sentiment_scores, file)

# Save sentiment word lists
with open('positive_words.pkl', 'wb') as file:
    pickle.dump(positive_words, file)

with open('negative_words.pkl', 'wb') as file:
    pickle.dump(negative_words, file)

with open('neutral_words.pkl', 'wb') as file:
    pickle.dump(neutral_words, file)

# Save word frequency distributions
with open('word_freq_positive.pkl', 'wb') as file:
    pickle.dump(word_freq_positive, file)

with open('word_freq_negative.pkl', 'wb') as file:
    pickle.dump(word_freq_negative, file)

with open('word_freq_neutral.pkl', 'wb') as file:
    pickle.dump(word_freq_neutral, file)
```

MODEL LOADING



```python
import pickle

# Load lemmatized words
with open('words_lemmatized.pkl', 'rb') as file:
    words_lemmatized = pickle.load(file)

# Load sentiment scores
with open('sentiment_scores.pkl', 'rb') as file:
    sentiment_scores = pickle.load(file)

# Load sentiment word lists
with open('positive_words.pkl', 'rb') as file:
    positive_words = pickle.load(file)

with open('negative_words.pkl', 'rb') as file:
    negative_words = pickle.load(file)

with open('neutral_words.pkl', 'rb') as file:
    neutral_words = pickle.load(file)

# Load word frequency distributions
with open('word_freq_positive.pkl', 'rb') as file:
    word_freq_positive = pickle.load(file)

with open('word_freq_negative.pkl', 'rb') as file:
    word_freq_negative = pickle.load(file)

with open('word_freq_neutral.pkl', 'rb') as file:
    word_freq_neutral = pickle.load(file)
```