## ***Developer Information***

**Name:** *Meena Reddimpalli*

***Email***: *reddimpalli575@gmail.com*

***Contact Number*****:** 9676058805

***Contributions****: Implemented Amazon Reviews Sentiment Analysis System by using built in python libraries and streamlit for web implementation.*

# Sentiment Analysis on Amazon Reviews

## 1. Introduction:

The purpose of this project is to scrape data from web pages containing Amazon reviews and perform sentiment analysis on the reviews. The scraped data will be stored in a Google Sheet, and the sentiment analysis results will be displayed in a Streamlit app.

## 2. Technologies Used:

- Python: The programming language used for writing the web scraping and sentiment analysis scripts.

- Pandas: A popular Python library for data manipulation and analysis.

- BeautifulSoup: A Python library for parsing HTML and XML documents.

- Google Sheets API: An API provided by Google for interacting with Google Sheets.

- Streamlit: A Python library for creating web apps.

- vaderSentiment: A Python library for sentiment analysis.

## 3. Web Scraping Process:

The web scraping process involves extracting data from HTML files containing Amazon reviews. The BeautifulSoup library is used to parse the HTML and extract relevant information such as customer names, ratings, review titles, and review bodies. The extracted data is then stored in a Pandas DataFrame. Duplicate customer names are dropped from the DataFrame to avoid redundancy.

The code provided is a Python script for web scraping Amazon reviews from HTML files and inserting the data into a Google Sheets spreadsheet using the Google Sheets API.

## The script makes use of the following libraries:

- `pandas` for data manipulation and analysis.

- `BeautifulSoup` (from `bs4`) for parsing HTML content.

- `re` for regular expressions to extract specific text patterns.

- `google_auth_oauthlib.flow` and `googleapiclient.discovery` for interacting with the Google Sheets API.

## Here is a breakdown of the code:

**1. Import the necessary libraries:**

```
import pandas as pd
from bs4 import BeautifulSoup as bs
import re
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
```

**2. Define a function `extract_data_from_html` to extract data from a single HTML file:**

```
def extract_data_from_html(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        content = file.read()
    soup = bs(content, 'html.parser')
    # Extract data
    cust_name = []
    ratings = []
```

```
reviews = []
review_body = []
# Find customer names
for span in soup.find_all('span', class_='a-profile-name'):
    try:
        cust_name.append(span.get_text())
    except:
        cust_name.append("")
# Find ratings and extract numeric part
for span in soup.find_all('span', class_='a-icon-alt'):
    try:
        rating_text = span.get_text()
        rating = re.search(r'(\d+\.\d+)', rating_text).group(1)
        ratings.append(rating)
    except:
        ratings.append("")
# Find review titles
for a in soup.find_all('a', class_='review-title-content'):
    try:
        span = a.find('span', class_='a-letter-space')
        if span:
            reviews.append(span.find_next_sibling('span').
            get_text())
        else:
```

```
            reviews.append("")
    except:
        reviews.append("")
# Find review bodies
for span in soup.find_all('span', {"data-hook": "review-
body"}):
    try:
        review_body.append(span.get_text().strip())
    except:
        review_body.append("")
# Ensure all lists have the same length
min_length = min(len(cust_name), len(ratings), len(reviews),
len(review_body))
cust_name = cust_name[:min_length]
ratings = ratings[:min_length]
reviews = reviews[:min_length]
review_body = review_body[:min_length]
# Create DataFrame
df = pd.DataFrame({
    'Customer Name': cust_name,
    'Ratings': ratings,
    'Reviews': reviews,
    'Review Body': review_body
})
```

```
    # Drop duplicate customer names
    df.drop_duplicates(subset=['Customer Name'], inplace=True)
    return df
```

**3. Define a list of HTML file paths and an empty list `data_frames` to hold all dataframes:**

```
html_files = ['Amazon1.html', 'Amazon2.html', 'Amazon3.html', ..., 'Amazon10.html']
data_frames = []
```

**4. Process each HTML file using the `extract_data_from_html` function and append the resulting dataframe to `data_frames`:**

```
for file_path in html_files:
    df = extract_data_from_html(file_path)
    data_frames.append(df)
```

**5. Concatenate all dataframes into a single dataframe `all_data`:**

```
all_data = pd.concat(data_frames, ignore_index=True)
```

**6. Convert the dataframe `all_data` to a list of lists `data`:**

```
data = all_data.values.tolist()
```

**7. Configure the Google Sheets API for authentication:**

```
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']
flow = InstalledAppFlow.from_client_secrets_file("key1.json", SCOPES)
cred = flow.run_local_server(port=0)
service = build('sheets', 'v4', credentials=cred)
```

#Make sure to replace `"key1.json"` with the path to your Google API key file.

**8. Update the spreadsheet with the data using the Google Sheets API:**

```
spreadsheet_id = "1629g6F4qSzTs2-MLm9TjbbKPnclK58BGw-hC-ybhJhs"
range_name = 'Sheet1'
body = {
   "values": data
}
response = service.spreadsheets().values().append(
   spreadsheetId=spreadsheet_id,
   range=range_name,
   valueInputOption="USER_ENTERED",
   insertDataOption="INSERT_ROWS",
   body=body
).execute()
print(response)  # Print the API response
```

Make sure to replace `spreadsheet_id` with the ID of your Google Sheets spreadsheet and `range_name` with the name of the sheet where you want to insert the data.

# Web-scraping.py

*web-scraping.py - C:\Projects\sentiments\Scripts\web-scraping.py (3.11.7)*

File Edit Format Run Options Window Help

```
import pandas as pd
from bs4 import BeautifulSoup as bs
import re
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build

# Function to extract data from a single HTML file
def extract_data_from_html(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        content = file.read()

    soup = bs(content, 'html.parser')

    # Extract data
    cust_name = []
    ratings = []
    reviews = []
    review_body = []

    # Find customer names
    for span in soup.find_all('span', class_='a-profile-name'):
        try:
            cust_name.append(span.get_text())
        except:
            cust_name.append("")

    # Find ratings and extract numeric part
    for span in soup.find_all('span', class_='a-icon-alt'):
        try:
            rating_text = span.get_text()
            rating = re.search(r'(\d+\.\d+)', rating_text).group(1)
            ratings.append(rating)
        except:
            ratings.append("")

    # Find review titles
    for a in soup.find_all('a', class_='review-title-content'):
        try:
            span = a.find('span', class_='a-letter-space')
            if span:
                reviews.append(span.find_next_sibling('span').get_text())
            else:
                reviews.append("")
        except:
            reviews.append("")

    # Find review bodies
    for span in soup.find_all('span', {"data-hook": "review-body"}):
        try:
            review_body.append(span.get_text().strip())
        except:
            review_body.append("")
```

*web-scraping.py - C:\Projects\sentiments\Scripts\web-scraping.py (3.11.7)*

File Edit Format Run Options Window Help

```
    # Ensure all lists have the same length
    min_length = min(len(cust_name), len(ratings), len(reviews), len(review_body))
    cust_name = cust_name[:min_length]
    ratings = ratings[:min_length]
    reviews = reviews[:min_length]
    review_body = review_body[:min_length]
    # Create DataFrame
    df = pd.DataFrame({
        'Customer Name': cust_name,
        'Ratings': ratings,
        'Reviews': reviews,
        'Review Body': review_body
    })

    # Drop duplicate customer names
    df.drop_duplicates(subset=['Customer Name'], inplace=True)

    return df

# List of HTML file paths
html_files = [ 'Amazon1.html','Amazon2.html']  # Add more file paths as needed
# Initialize an empty list to hold all dataframes
data_frames = []
# Process each HTML file and append the data to the list
for file_path in html_files:
    df = extract_data_from_html(file_path)
    data_frames.append(df)

# Concatenate all dataframes
all_data = pd.concat(data_frames, ignore_index=True)
# Convert DataFrame to a list of lists
data = all_data.values.tolist()
# Add column headers to data
#headers = ["Customer Name", "Ratings", "Reviews", "Review Body"]
#data.insert(0, headers)
# Google Sheets API setup
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']
flow = InstalledAppFlow.from_client_secrets_file("key1.json", SCOPES)
cred = flow.run_local_server(port=0)
service = build('sheets', 'v4', credentials=cred)
# Update the spreadsheet with the data
spreadsheet_id = "1629g6F4qSzTs2-MLm9TjbbKPnclK58BGw-hC-ybhJhs"
range_name = 'Sheet1'  # Specify the sheet name

body = {
    "values": data
}
response = service.spreadsheets().values().append(
    spreadsheetId=spreadsheet_id,
    range=range_name,
    valueInputOption="USER_ENTERED",
    insertDataOption="INSERT_ROWS",
    body=body
).execute()
print(response)
```

## 4. Sentiment Analysis Process:

The sentiment analysis process is implemented using the vaderSentiment library. Each review is assigned a sentiment score based on its polarity, and sentiments are categorized as positive, negative, or neutral. The sentiment analysis results are added as a new column in the DataFrame.

**Necessary Libraries:**

- streamlit: for creating the Streamlit app.

- pandas: for data manipulation and analysis.

- google_auth_oauthlib.flow: for Google Sheets API authentication.

- googleapiclient.discovery: for interacting with Google Sheets API.

- vaderSentiment.vaderSentiment: for sentiment analysis using VADER.

- altair: for creating visualizations.

- wordcloud: for generating word clouds.

- matplotlib.pyplot: for displaying the word cloud.

- plotly.express: for creating interactive visualizations.

## Here is a breakdown of the code:

**1. Import the necessary libraries:**

```
import streamlit as st
import pandas as pd
from google_auth_oauthlib.flow import InstalledAppFlow
```

```
from googleapiclient.discovery import build

from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer

import altair as alt

from wordcloud import WordCloud

import matplotlib.pyplot as plt

import plotly.express as px
```

**2. Define the scope for the Google Sheets API.**

```
SCOPES = ["https://www.googleapis.com/auth/spreadsheets"]
```

**3. Define a function to get the Google Sheets service using OAuth2 authentication.**

```
 @st.cache_resource

def get_google_sheet_service():

   flow = InstalledAppFlow.from_client_secrets_file("key.json",
SCOPES)

   cred = flow.run_local_server(port=0)

   return build("sheets", "v4",
credentials=cred).spreadsheets().values()
```

**4. Define a function to fetch data from Google Sheets.**

```
def get_sheet_data(service, spreadsheet_id, range_name,
retries=3):

   for attempt in range(retries):

      try:

         result = service.get(spreadsheetId=spreadsheet_id,

         range=range_name).execute()
```

```
            return result.get('values', [])
        except socket.gaierror as e:
            if attempt < retries - 1:
                time.sleep(2)  # Wait before retrying
            else:
                st.error(f"Network error: {e}")
                return []
```

**5. Define a function to perform sentiment analysis on the reviews.**

```
def perform_sentiment_analysis(df):
    analyzer = SentimentIntensityAnalyzer()
    sentiments = []
    for review in df["Reviews"]:
        sentiment_score =
analyzer.polarity_scores(review)['compound']
        if sentiment_score > 0.5:
            sentiments.append("Positive 😀 ")
        elif sentiment_score < -0.5:
            sentiments.append("Negative 😠 ")
        else:
            sentiments.append("Neutral 😐 ")
    df['Sentiments'] = sentiments
    return df
```

**6. Define a function to update Google Sheets with the new data.**

```
def update_google_sheet(service, spreadsheet_id, data):
```

```
    body = {"values": data}
    service.update(
        spreadsheetId=spreadsheet_id,
        range="A:E",
        valueInputOption="USER_ENTERED",
        body=body
    ).execute()
```

**7. Define the main Streamlit app function.**

```
def main():
    # Title of the app
    st.title("Amazon Review Sentiment Analysis")
    # Display an image
    image = "https://www.revuze.it/blog/wp-
    content/uploads/sites/2/2020/03/Amazon-Review-
    Analysis.png"
    st.markdown(
        f'<div style="text-align:center;">'
        f'<img src="{image}" style="width:500px">'
        '</div>',
        unsafe_allow_html=True
    )

    # Custom CSS for button and text input
    st.markdown(
```

```
"""
<style>
.stButton button {
    background-color: #00CCCC;
    color: black;
}
.stTextInput input {
    color: white;
}
</style>
""",
unsafe_allow_html=True
)

# Input fields for Google Sheets ID and range
spreadsheet_id = st.text_input("Enter Google Sheets ID",
"1629g6F4qSzTs2-MLm9TjbbKPnclK58BGw-hC-ybhJhs")
range_name = st.text_input("Enter Range Name", "A:D")
if st.button("Analyze Sentiments"):
    st.write("Fetching data from Google Sheets...")

    # Fetch data from Google Sheets
    service = get_google_sheet_service()
    data = get_sheet_data(service, spreadsheet_id,
```

```
    range_name)
    if not data:
        st.write("No data found.")
        return
    # Convert data to DataFrame and perform sentiment
    analysis
    df = pd.DataFrame(data[1:], columns=data[0])
    df = perform_sentiment_analysis(df)
    updated_data = [df.columns.tolist()] + df.values.tolist()
    # Update Google Sheets with the new data
    update_google_sheet(service, spreadsheet_id,
    updated_data)
    st.write("Updated DataFrame:")
    # Function to apply background color based on sentiment
      values
    def apply_sentiment_color(val):
        if val == 'Positive 😀 ':
            return 'background-color: yellow; color: black'
        elif val == 'Negative 😡 ':
            return 'background-color: red; color: white'
        else:
            return 'background-color: white; color: black'
    # Apply styling to DataFrame
    styled_df = df.style.applymap(apply_sentiment_color,
```

```
subset=['Sentiments'])
# Apply header styling
header_style = {
    'selector': 'th',
    'props': [('background-color', 'blue'), ('color', 'white')]
}
# Display styled DataFrame
st.write(styled_df.set_table_styles([header_style]),
unsafe_allow_html=True)
# Display sentiment analysis results
st.title("Sentiment Analysis Results:")
# Create a bar chart for sentiment counts
sentiment_counts =
df['Sentiments'].value_counts().reset_index()
sentiment_counts.columns = ['Sentiment', 'Count']
col1, col2 = st.columns(2)
with col1:
    bar_chart =
    alt.Chart(sentiment_counts).mark_bar().encode(
        x='Sentiment',
        y='Count',
        color='Sentiment'
    ).properties(
        width=300,
```

```
            height=300,
            title="Sentiment Count Bar Chart"
        )
        st.altair_chart(bar_chart)
    with col2:
        st.subheader("Pie Chart")
        counts = df['Sentiments'].value_counts()
        fig_pie = px.pie(values=counts, names=counts.index)
        fig_pie.update_layout(
            width=300,
            height=300
        )
        st.plotly_chart(fig_pie)
    # Display a word cloud of the reviews
    st.subheader("Word Cloud of Reviews:")
    text = " ".join(review for review in df['Reviews'])
    wordcloud = WordCloud(width=800, height=400,
    background_color='white').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    st.pyplot(plt)
if __name__ == "__main__":
    main()
```

# Amazon-review.py

*Amazon-review.py - C:\Projects\sentiments\Scripts\Amazon-review.py (3.11.7)*

File Edit Format Run Options Window Help

```
import streamlit as st
import pandas as pd
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import altair as alt
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import plotly.express as px

# Define the scope for Google Sheets API
SCOPES = ["https://www.googleapis.com/auth/spreadsheets"]
# Function to get Google Sheets service
@st.cache_resource
def get_google_sheet_service():
    flow = InstalledAppFlow.from_client_secrets_file("key.json", SCOPES)
    cred = flow.run_local_server(port=0)
    return build("sheets", "v4", credentials=cred).spreadsheets().values()

# Function to fetch data from Google Sheets
def get_sheet_data(service, spreadsheet_id, range_name, retries=3):
    for attempt in range(retries):
        try:
            result = service.get(spreadsheetId=spreadsheet_id, range=range_name).execute()
            return result.get('values', [])
        except socket.gaierror as e:
            if attempt < retries - 1:
                time.sleep(2)  # Wait before retrying
            else:
                st.error(f"Network error: {e}")
                return []

# Function to perform sentiment analysis on reviews
def perform_sentiment_analysis(df):
    analyzer = SentimentIntensityAnalyzer()
    sentiments = []
    for review in df["Reviews"]:
        sentiment_score = analyzer.polarity_scores(review)['compound']
        if sentiment_score > 0.5:
            sentiments.append("Positive😀")
        elif sentiment_score < -0.5:
            sentiments.append("Negative😞")
        else:
            sentiments.append("Neutral😐")
    df['Sentiments'] = sentiments
    return df
```

*Amazon-review.py - C:\Projects\sentiments\Scripts\Amazon-review.py (3.11.7)*

File Edit Format Run Options Window Help

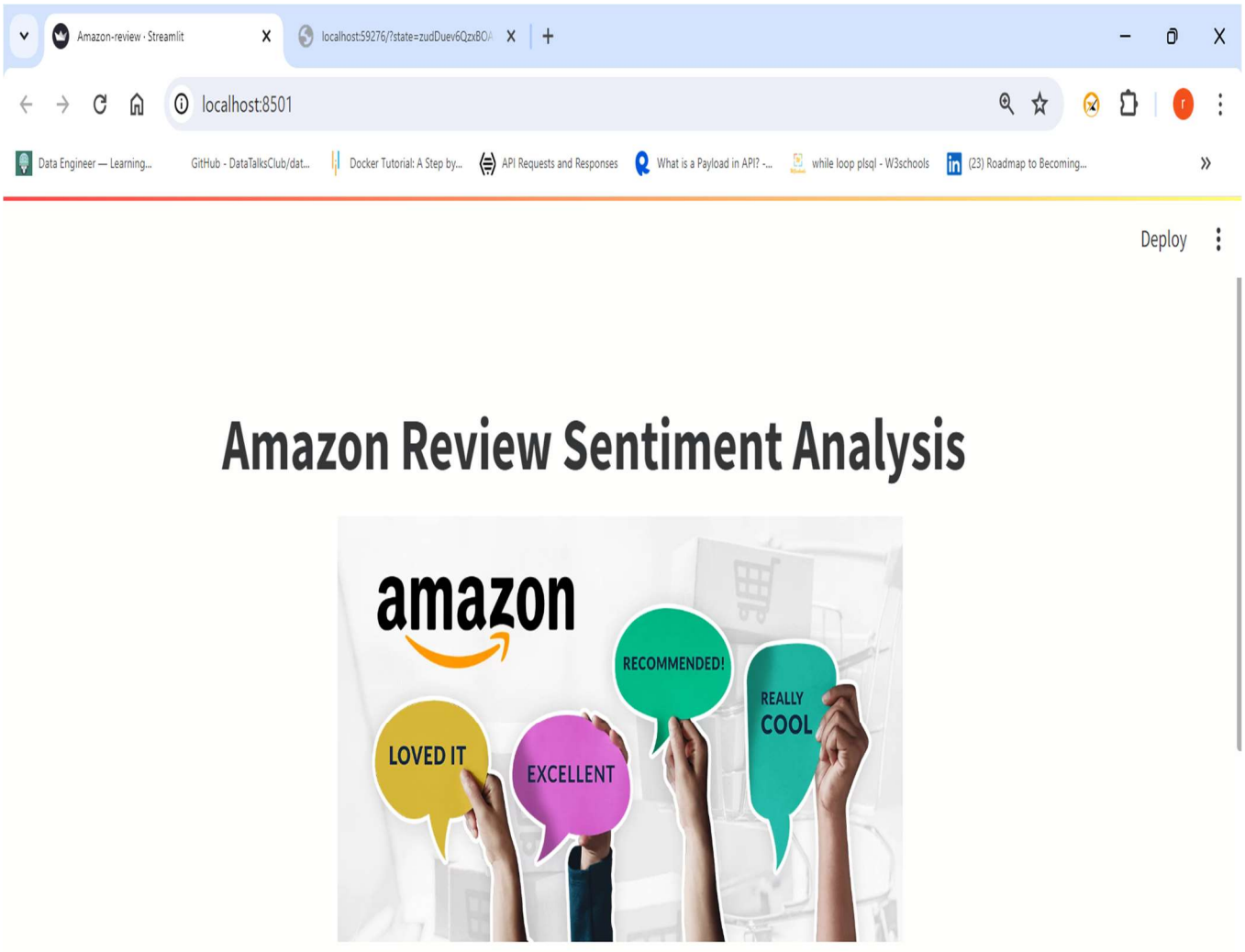
```
# Function to update Google Sheets with the new data
def update_google_sheet(service, spreadsheet_id, data):
    body = {"values": data}
    service.update(
        spreadsheetId=spreadsheet_id,
        range="A:E",
        valueInputOption="USER_ENTERED",
        body=body
    ).execute()

# Streamlit app
def main():
    st.title("Amazon Review Sentiment Analysis")  # Title of the app

    # Display an image
    image = "https://www.revuze.it/blog/wp-content/uploads/sites/2/2020/03/Amazon-Review-Analysis.png"
    st.markdown(
        f'<div style="text-align:center;">'
        f'<img src="{image}" style="width:500px">'
        '</div>',
        unsafe_allow_html=True
    )

    # Custom CSS for button and text input
    st.markdown(
    """
    <style>
    .stButton button {
        background-color: #00CCCC;
        color: black;
    }
    .stTextInput input {
        color: white;
    }
    </style>
    """,
    unsafe_allow_html=True
    )

    # Input fields for Google Sheets ID and range
    spreadsheet_id = st.text_input("Enter Google Sheets ID", "1629g6F4qSzTs2-MLm9TjbbKPnclK58BGw-hC-ybhJhs")
    range_name = st.text_input("Enter Range Name", "A:D")

    if st.button("Analyze Sentiments"):
        st.write("Fetching data from Google Sheets...")

        # Fetch data from Google Sheets
        service = get_google_sheet_service()
        data = get_sheet_data(service, spreadsheet_id, range_name)

        if not data:
            st.write("No data found.")
            return
```

*Amazon-review.py - C:\Projects\sentiments\Scripts\Amazon-review.py (3.11.7)*

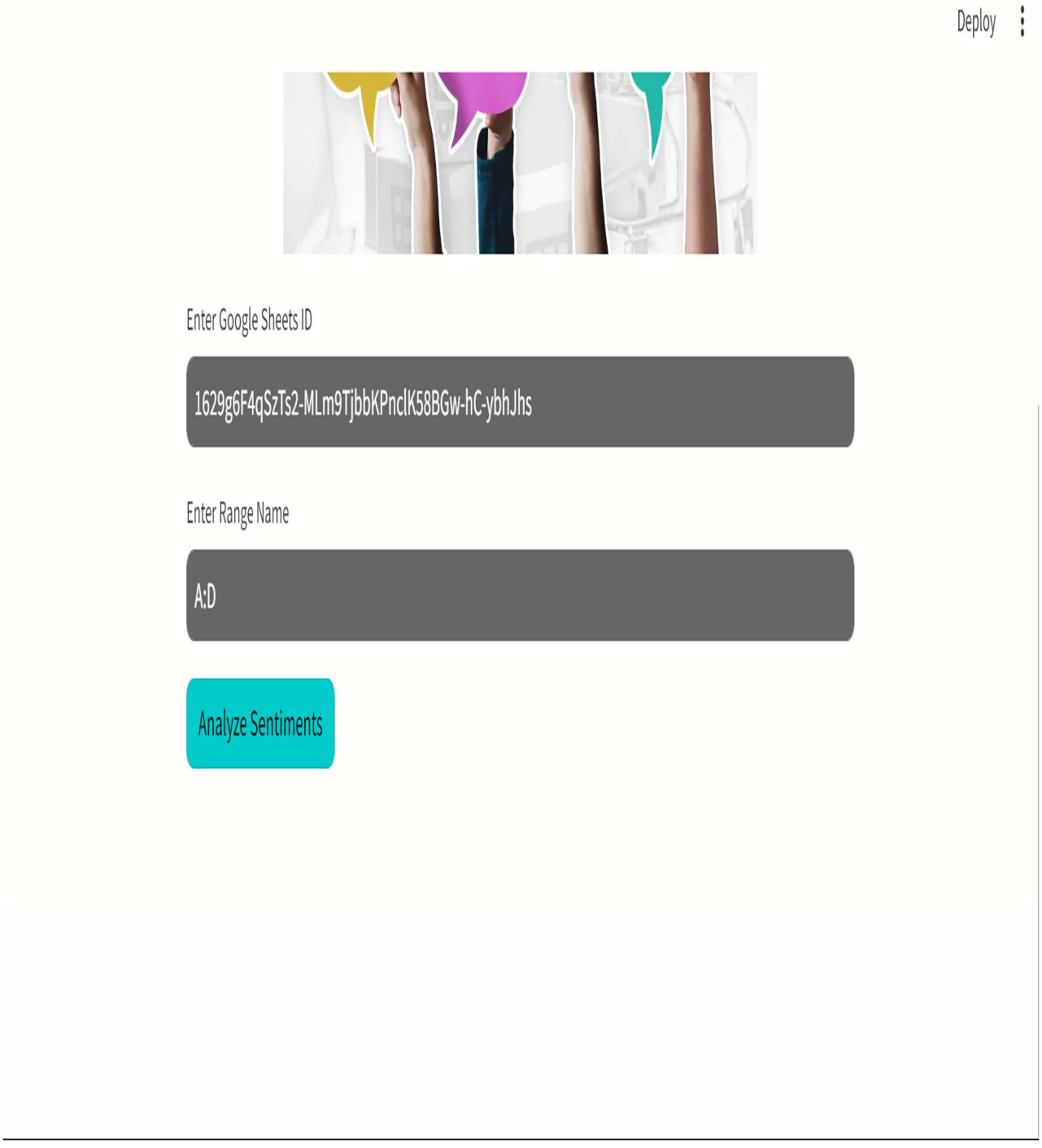
File Edit Format Run Options Window Help

```
# Convert data to DataFrame and perform sentiment analysis
df = pd.DataFrame(data[1:], columns=data[0])
df = perform_sentiment_analysis(df)
updated_data = [df.columns.tolist()] + df.values.tolist()

# Update Google Sheets with the new data
update_google_sheet(service, spreadsheet_id, updated_data)
st.write("Updated DataFrame:")

# Function to apply background color based on sentiment values
def apply_sentiment_color(val):
    if val == 'Positive😀':
        return 'background-color: yellow; color: black'
    elif val == 'Negative😞':
        return 'background-color: red; color: white'
    else:
        return 'background-color: white; color: black'

# Apply styling to DataFrame
styled_df = df.style.applymap(apply_sentiment_color, subset=['Sentiments'])

# Apply header styling
header_style = {
    'selector': 'th',
    'props': [('background-color', 'blue'), ('color', 'white')]
}

# Display styled DataFrame
st.write(styled_df.set_table_styles([header_style]), unsafe_allow_html=True)

# Display sentiment analysis results
st.title("Sentiment Analysis Results:")

# Create a bar chart for sentiment counts
sentiment_counts = df['Sentiments'].value_counts().reset_index()
sentiment_counts.columns = ['Sentiment', 'Count']
col1, col2 = st.columns(2)
with col1:
    bar_chart = alt.Chart(sentiment_counts).mark_bar().encode(
        x='Sentiment',
        y='Count',
        color='Sentiment'
    ).properties(
        width=300,
        height=300,
        title="Sentiment Count Bar Chart"
    )
    st.altair_chart(bar_chart)
```

```
        with col2:
            st.subheader("Pie Chart")
            counts = df['Sentiments'].value_counts()
            fig_pie = px.pie(values=counts, names=counts.index)
            fig_pie.update_layout(
                width=300,
                height=300
            )
            st.plotly_chart(fig_pie)

        # Display a word cloud of the reviews
        st.subheader("Word Cloud of Reviews:")
        text = " ".join(review for review in df['Reviews'])
        wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
        plt.figure(figsize=(10, 5))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis("off")
        st.pyplot(plt)

if __name__ == "__main__":
    main()
```




*Amazon Review Sentiment Analysis Web Application Inter face*




***Here we can enter the Google Sheets ID and Range_Name according to google sheet and press the button to analyse the Sentiment of 'Reviews' column.***

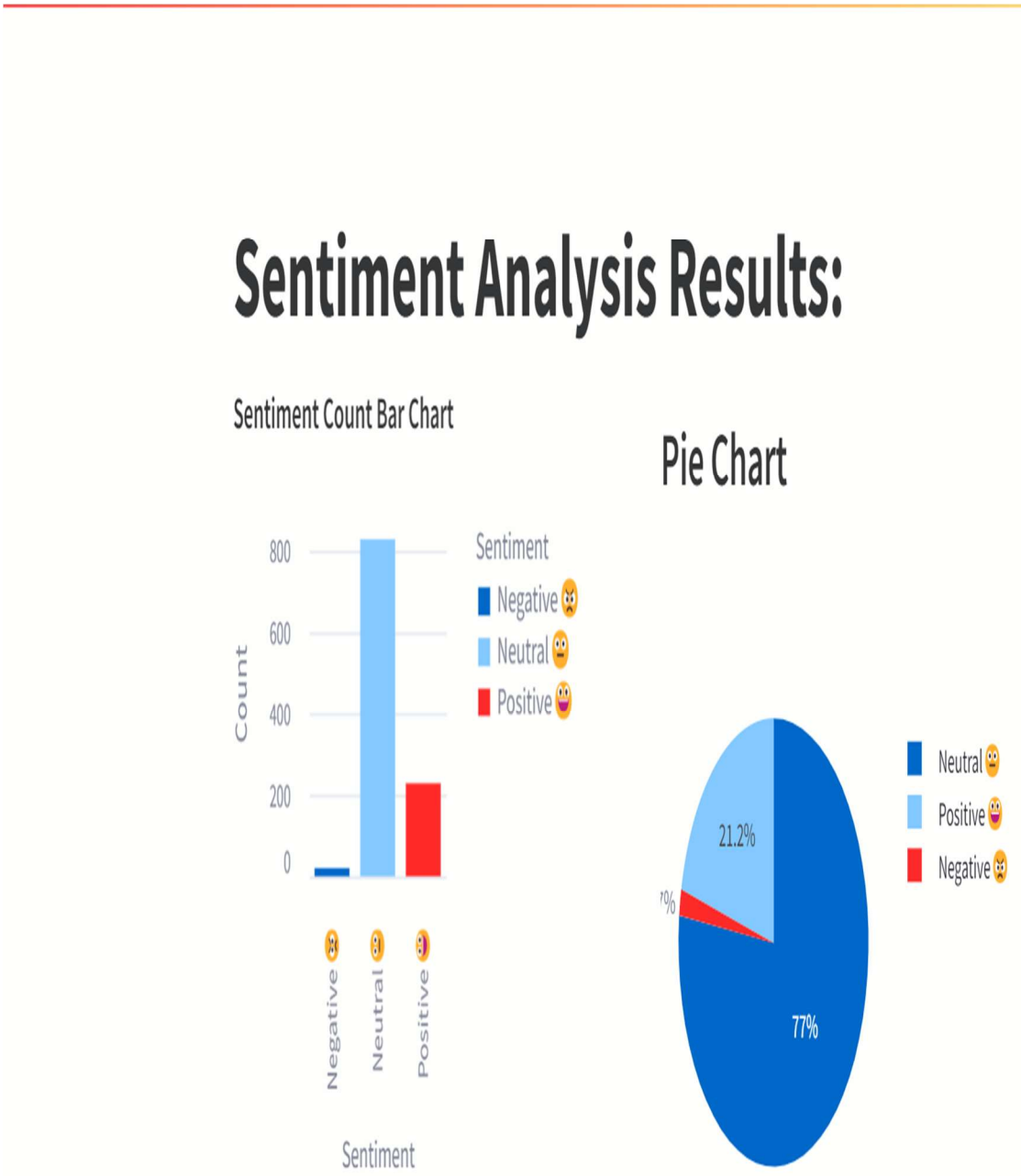
Deploy

Fetching data from Google Sheets...

Updated DataFrame:

| | Customer Name | Ratings | Reviews | Review Body | Sentiments |
|---|---|---|---|---|---|
| 0 | Anusuya Singh Rana | 4.1 | Amazing Phone on th | Best phone under 10k. No problem with | Positive |
| 1 | Placeholder | 5 | Good one | Nice for a dad or mom for daily use if u a | Neutral |
| 2 | kiranklg | 5 | Very nice | All good | Neutral |
| 3 | RobbyJK | 4 | Very bad product and | ProductVery Very Very bad productSabs | Negative |
| 4 | Bishnusarkar | 5 | Phone is good for da | Phone is good for day to day activities n | Neutral |
| 5 | Durga Kalyan | 1 | Low cost | But other product give so many features | Neutral |
| 6 | Arpan Bhattacharya | 3 | Redmi 12(5G) | Good delivery I am happy | Neutral |
| 7 | Anusuya Singh Rana | 4.1 | Mobile | One of the best phone in mid range...tha | Neutral |
| 8 | Placeholder | 5 | Nice | Mi 12 5g it's good. | Neutral |

*After analysing the reviews it returns column with name ‘ Sentiments’ having Positive(Yellow),Negative(Red) and Neutral(White) analysing values.*




*Here is the bar chart for Sentiments (Positive ,Negative and Neutral) count and Pie chart for percentage of sentiments.*


*word cloud, is a visual representation of text data where the size of each word indicates its frequency or importance within the source text.*

## 5. Conclusion:

This project demonstrates the integration of web scraping, sentiment analysis, and Google Sheets API in a Streamlit app. It provides a convenient way to scrape Amazon reviews, analyze the sentiments of the reviews, and visualize the results in an interactive web interface.