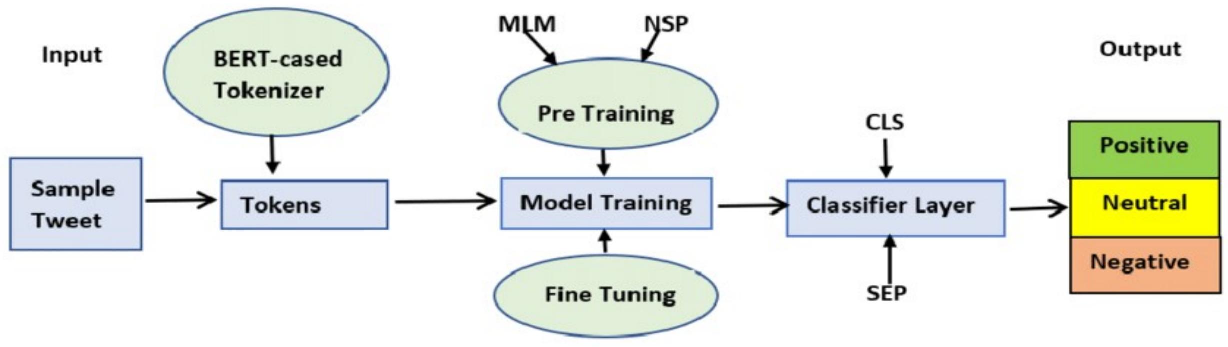
Sentiment Analysis for Marketing
PHASE -3

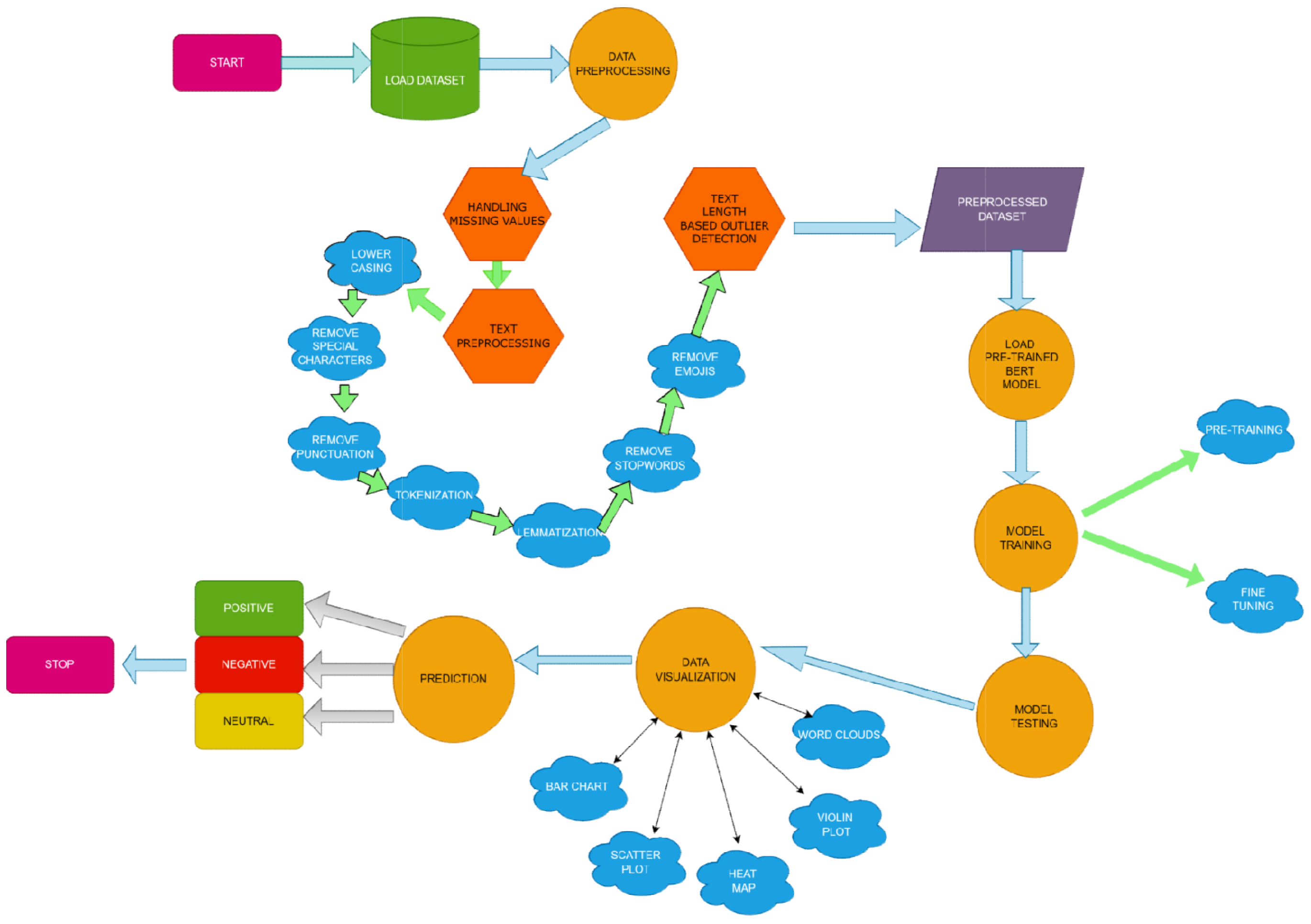
DATE	17 October 2023
TEAM ID	Proj-212173-Team-1
PROJECT NAME	Sentiment analysis for Marketing

Sentimental Analysis Architecture Model:

Simplified Model:



Brief Model:



DATA VISUALIZATION:

Data visualization is the graphical representation of data to help people understand and interpret the information contained within it. It involves creating visual representations, such as charts, graphs, maps, and dashboards, to present data in a way that is visually appealing, informative, and accessible. The primary goal of data visualization is to make complex data more understandable, revealing patterns, trends, and insights that may not be apparent from raw data alone.

Here are key aspects of data visualization:

1. Data Presentation: Data visualization transforms data into visual elements like lines, bars, points, shapes, colors, and text. These elements convey information more effectively than rows and columns of numbers.

2. Understanding Complex Data: Visualization simplifies complex data, enabling users to grasp information quickly and make data-driven decisions.

3. Revealing Patterns and Trends: Visualization can highlight patterns, trends, outliers, and correlations in the data, making it easier to draw conclusions and insights.

4. Communication: Data visualization is a powerful tool for communicating data and findings to a broad audience. It helps convey information in a way that is accessible to both technical and non-technical stakeholders.

5. Exploration and Discovery: Visualization can facilitate data exploration by allowing users to interact with data, zoom in on specific details, or filter data to discover hidden insights.

6. Decision-Making: Data visualizations are valuable for decision-making processes, as they provide a clear and concise representation of data that can support informed choices.

Common types of data visualizations include:

1. Word Clouds: Word clouds visually represent the most frequently occurring words in a dataset, with word size indicating frequency. They provide an intuitive summary of the most common words in text data, making them a useful tool for identifying key sentiments and themes.

2. Bar Charts: Bar charts can display the distribution of sentiment categories (e.g., positive, negative, neutral) in a dataset. Each sentiment category is represented by a bar, and the height of the bar corresponds to the number of occurrences, providing a clear view of sentiment distribution.

3. Pie Charts: Pie charts are another way to visualize sentiment distribution. Each sentiment category is represented as a slice of the pie, with the size of the slice proportional to the percentage of each sentiment in the dataset.

4. Heatmaps: Heatmaps can be used to visualize the sentiment of text data over time or across different categories. They help identify patterns and trends in sentiment changes.

5. Line Charts: Line charts can show sentiment trends over time. For example, you can track how sentiment about a product or service evolves over weeks or months based on user reviews or social media posts.

6. Scatter Plots: If you have numerical sentiment scores associated with text data, scatter plots can display the relationship between sentiment scores and other variables, such as time or word count.

7. Violin Plots: Violin plots can display the distribution of sentiment scores by category or group. They show the shape of the distribution, including the median and quartiles, and can help identify differences in sentiment among groups.

8. Network Graphs: Network graphs can reveal relationships and connections between entities or terms mentioned in text data, allowing you to explore how sentiment is related to specific topics or entities.

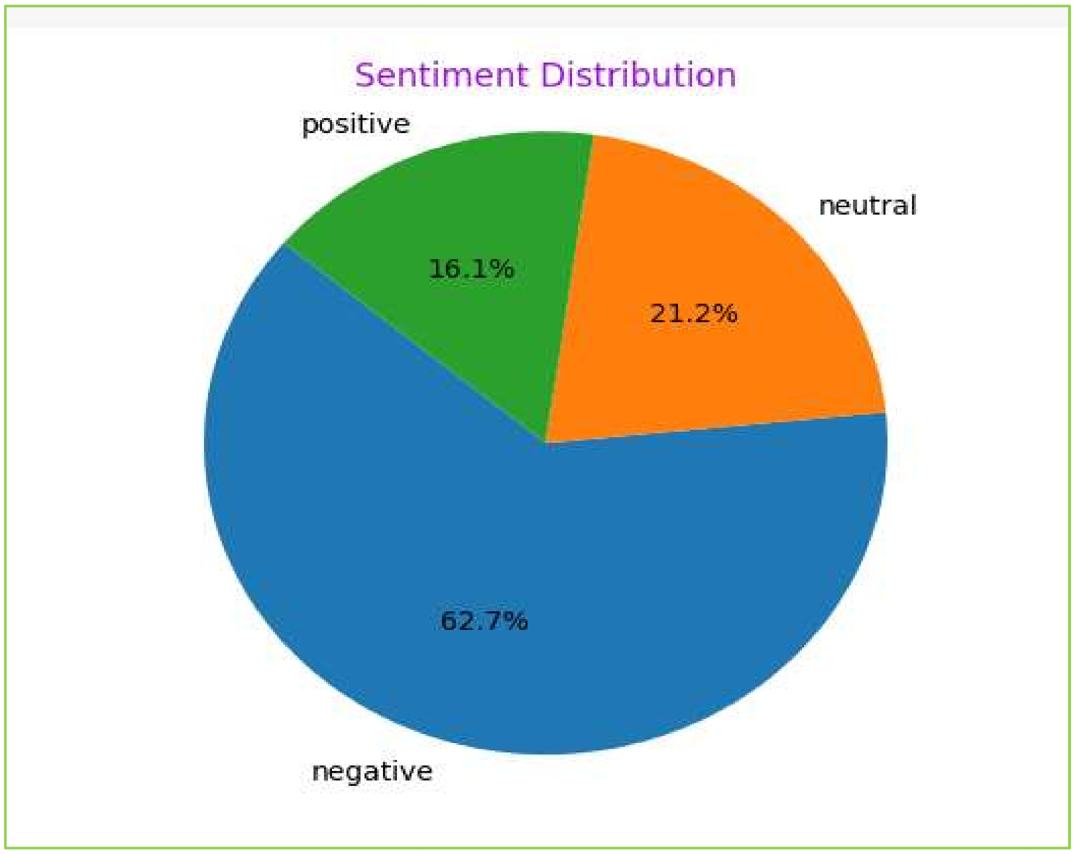
9. Dashboard: Dashboards provide an interactive way to explore sentiment data. They can combine multiple visualizations, filters, and controls to enable in-depth analysis.

10. Geospatial Visualization: If sentiment data is associated with geographic locations, maps can be used to show the distribution of sentiments across regions.

CODE:

```
sentiment_counts = df['airline_sentiment'].value_counts()
labels = sentiment_counts.index
sizes = sentiment_counts.values
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.axis('equal')
plt.title("Sentiment Distribution",color='#a114de')
plt.show()
```

OUTPUT:



EXPLANATION:

sentiment_counts = df['airline_sentiment'].value_counts(): This code calculates the counts of unique sentiment categories in the 'airline_sentiment' column of your DataFrame (`df`) and stores the counts in the `sentiment_counts` variable.

plt.figure(figsize=(8, 6)): This line sets the figure size of the plot to 8 inches in width and 6 inches in height, determining the dimensions of the plot.

sentiment_counts.plot(kind='bar', color=['red', 'yellow', 'green']): This line creates a bar plot with sentiment categories on the x-axis and their corresponding counts on the y-axis. The bars are colored with 'red' for negative, 'yellow' for neutral, and 'green' for positive sentiment.

plt.xlabel('Sentiment', color='#1da1a8'): This sets the label for the x-axis to 'Sentiment' and customizes the label text color to a specific shade of blue-green.

plt.ylabel('Count', color='#1da1a8'): This sets the label for the y-axis to 'Count' and customizes the label text color to the same shade of blue-green.

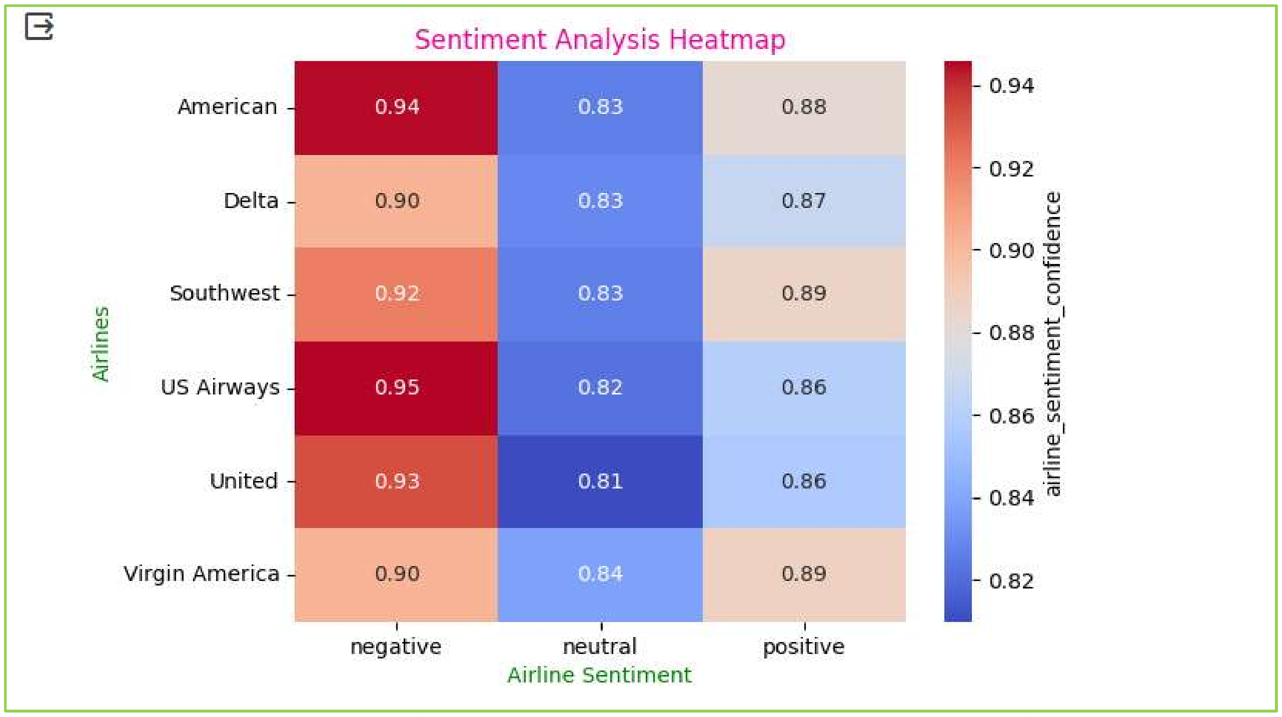
plt.title('Sentiment Analysis Results for Text Data', color='#a114de'): This sets the title of the plot to 'Sentiment Analysis Results for Text Data' and customizes the title text color to a specific shade of purple.

plt.show(): This line displays the bar plot in your Python environment.

CODE:

```
heatmap_data = df.pivot_table(index='airline', columns='airline_sentiment',
values='airline_sentiment_confidence', aggfunc='mean')
sns.heatmap(heatmap_data, cmap="coolwarm", annot=True, fmt=".2f", cbar_kws={'label':
'airline_sentiment_confidence'})
plt.xlabel('Airline Sentiment',color='green')
plt.ylabel('Airlines',color='green')
plt.title('Sentiment Analysis Heatmap',color='#e6079b')
plt.show()
```

OUTPUT:



EXPLANATION:

heatmap_data = df.pivot_table(index='airline', columns='airline_sentiment', values='airline_sentiment_confidence', aggfunc='mean'): This line pivots the data in your DataFrame (`df`). It calculates the mean value of 'airline_sentiment_confidence' for each combination of 'airline' and 'airline_sentiment' and arranges the results in a table format. This table is suitable for creating a heatmap.

sns.heatmap(heatmap_data, cmap="coolwarm", annot=True, fmt=".2f", cbar_kws={'label': 'airline_sentiment_confidence'}): This code generates a heatmap using the Seaborn library (`sns`). The parameters are as follows:

heatmap_data: The data to be visualized in the heatmap, which is the pivot table created in step 1.

cmap="coolwarm": Sets the color map to "coolwarm" for the heatmap, which represents values with colors ranging from cool (e.g., blue) to warm (e.g., red).

annot=True: Displays the actual values on the heatmap cells.

fmt=".2f": Formats the values in the cells to two decimal places.

cbar_kws={'label': 'airline_sentiment_confidence'}: Adds a color bar (legend) to the heatmap with the label 'airline_sentiment_confidence'.

plt.xlabel('Airline Sentiment', color='green'): This sets the label for the x-axis to 'Airline Sentiment' and customizes the label text color to green.

plt.ylabel('Airlines', color='green'): This sets the label for the y-axis to 'Airlines' and customizes the label text color to green.

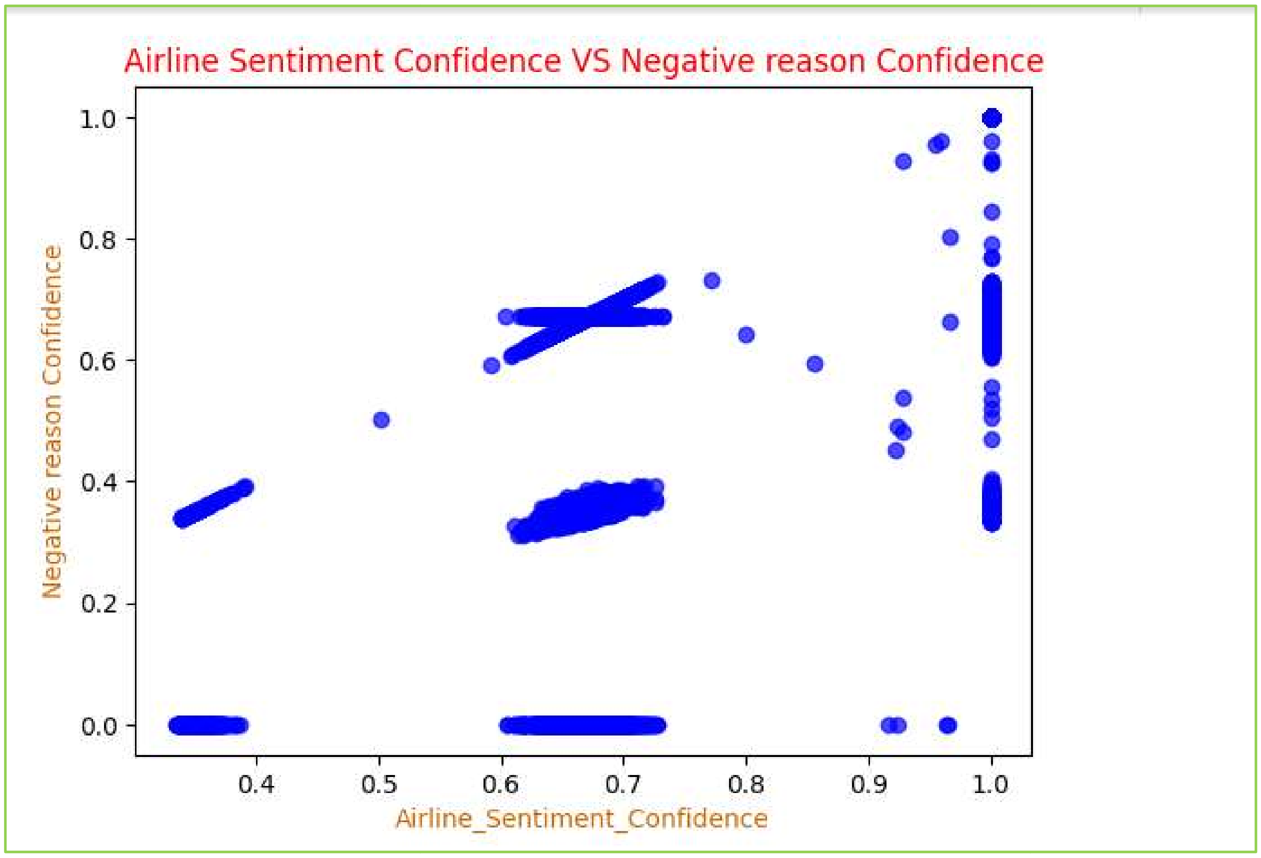
plt.title('Sentiment Analysis Heatmap', color='#e6079b'): This sets the title of the heatmap to 'Sentiment Analysis Heatmap' and customizes the title text color to a specific shade of pink.

plt.show(): This line displays the heatmap in your Python environment.

CODE:

```
x=df['airline_sentiment_confidence']
y=df['negativereason_confidence']
plt.scatter(x, y, marker='o', color='#06c992', alpha=0.7)
plt.xlabel('Airline_Sentiment_Confidence',color='#c96806')
plt.ylabel('Negative reason Confidence',color='#c96806')
plt.title('Airline Sentiment Confidence VS Negative reason Confidence',color='red')
plt.show()
```

OUTPUT:



EXPLANATION:

x = df['airline_sentiment_confidence']: This line extracts the 'airline_sentiment_confidence' values as `x`, which will be plotted on the x-axis.

y = df['negativereason_confidence']: This line extracts the 'negativereason_confidence' values as `y`, which will be plotted on the y-axis.

plt.scatter(x, y, marker='o', color='#06c992', alpha=0.7): This line creates a scatter plot with `x` as the x-values (airline sentiment confidence) and `y` as the y-values (negative reason confidence). The following parameters are used:

marker='o': Sets the marker style for data points as circles ('o').

color='#06c992': Sets the color of the data points to a custom shade of green.

alpha=0.7: Adjusts the transparency of the data points to 0.7, making them slightly transparent.

plt.xlabel('Airline_Sentiment_Confidence', color='#c96806'): This sets the label for the x-axis to 'Airline Sentiment Confidence' and customizes the label text color to a custom shade of orange.

plt.ylabel('Negative reason Confidence', color='#c96806'): This sets the label for the y-axis to 'Negative Reason Confidence' and customizes the label text color to the same shade of orange.

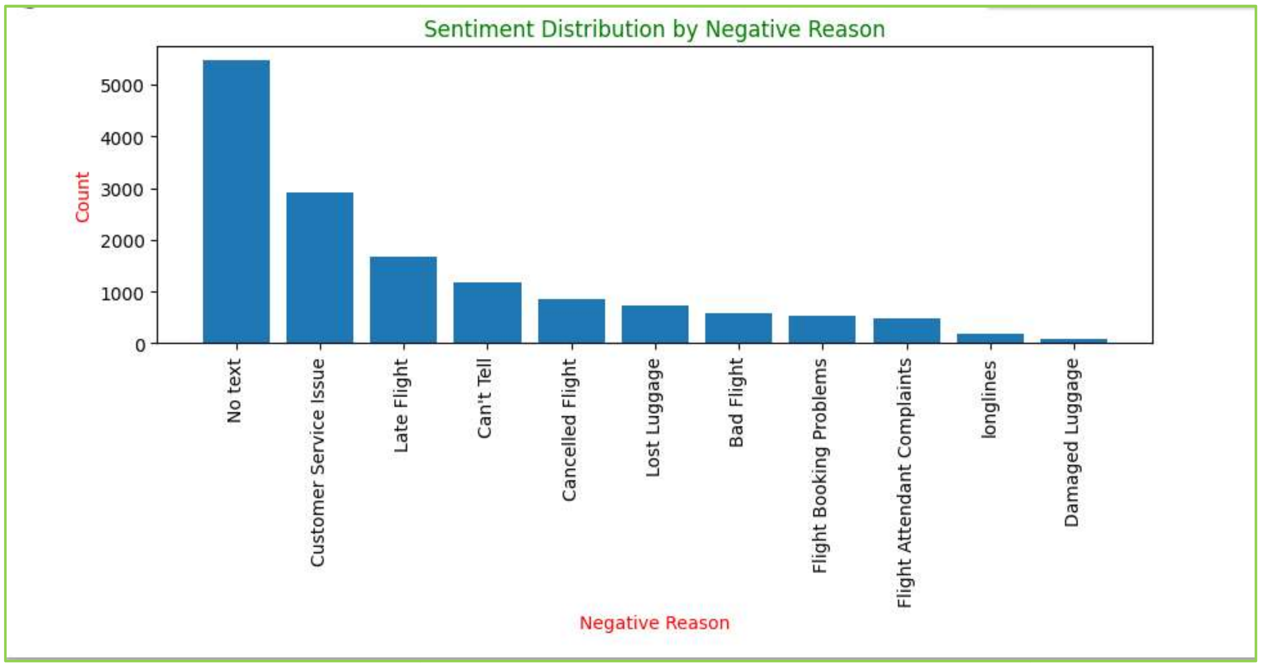
plt.title('Airline Sentiment Confidence VS Negative Reason Confidence', color='red'): This sets the title of the plot to 'Airline Sentiment Confidence VS Negative Reason Confidence' and customizes the title text color to red.

plt.show(): This line displays the scatter plot in your Python environment.

CODE:

```
negative_reason_counts = df['negativereason'].value_counts()
x = negative_reason_counts.index
y = negative_reason_counts.values
plt.figure(figsize=(10, 6))
plt.bar(x,y)
plt.xlabel('Negative Reason',color='red')
plt.ylabel('Count',color='red')
plt.title('Sentiment Distribution by Negative Reason',color='green')
plt.xticks(rotation=90)
plt.show()
```

OUTPUT:



EXPLANATION:

negative_reason_counts = df['negativereason'].value_counts(): This code calculates the counts of unique negative reasons in the 'negativereason' column of your DataFrame (`df`) and stores the counts in the `negative_reason_counts` variable.

x = negative_reason_counts.index: This line extracts the unique negative reasons as `x` (x-values) from the index of the `negative_reason_counts` Series.

y = negative_reason_counts.values: This line extracts the corresponding counts of each unique negative reason as `y` (y-values) from the values of the `negative_reason_counts` Series.

plt.figure(figsize=(10, 6)): This code sets the figure size of the plot to 10 inches in width and 6 inches in height, determining the dimensions of the plot.

plt.bar(x, y): This line creates a bar plot with `x` as the x-values (negative reasons) and `y` as the y-values (counts). The bars are plotted with the default colors.

plt.xlabel('Negative Reason', color='red'): This sets the label for the x-axis to 'Negative Reason' and customizes the label text color to 'red'.

plt.ylabel('Count', color='red'): This sets the label for the y-axis to 'Count' and customizes the label text color to 'red'.

plt.title('Sentiment Distribution by Negative Reason', color='green'): This sets the title of the plot to 'Sentiment Distribution by Negative Reason' and customizes the title text color to 'green'.

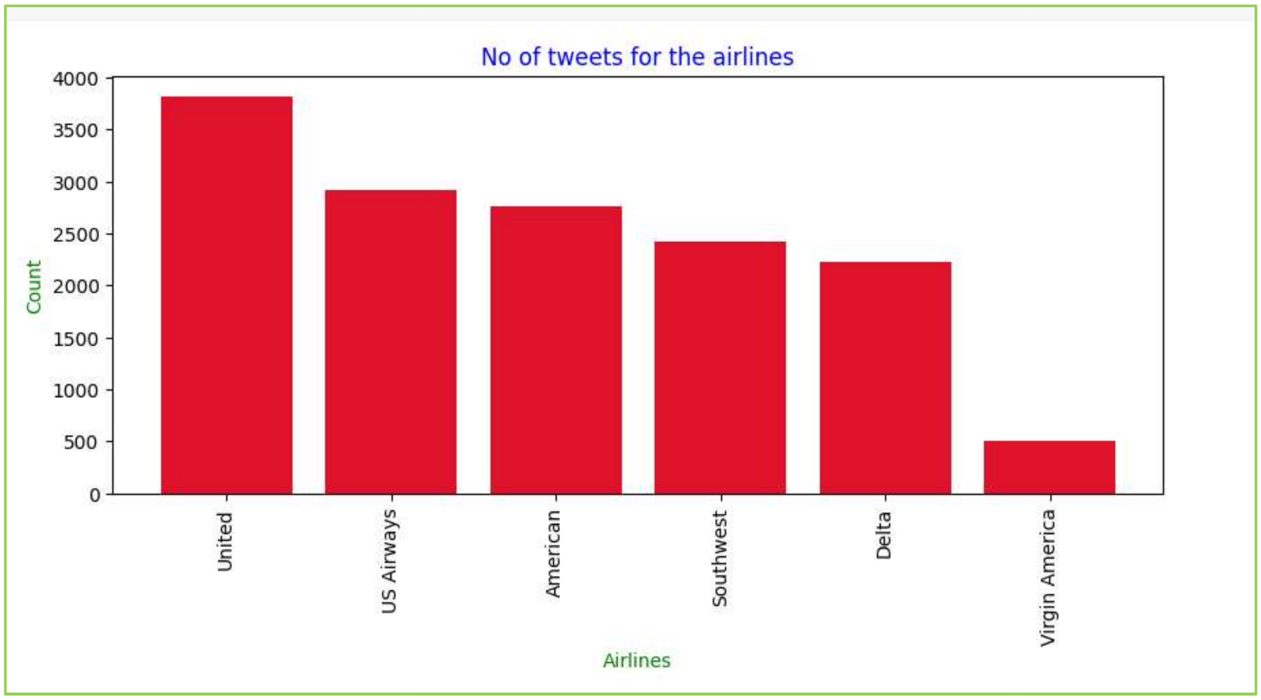
plt.xticks(rotation=90): This line rotates the x-axis labels (negative reasons) by 90 degrees to make them more readable when dealing with long labels.

plt.show(): This line displays the bar plot in your Python environment.

CODE:

```
airline_counts=df['airline'].value_counts()
x=airline_counts.index
y=airline_counts.values
plt.figure(figsize=(10, 4))
plt.bar(x,y,color='#de122a')
plt.xlabel('Airlines',color='green')
plt.ylabel('Count',color='green')
plt.title('No of tweets for the airlines',color='blue')
plt.xticks(rotation=90)
plt.show()
```

OUTPUT:



EXPLANATION:

airline_counts = df['airline'].value_counts(): This code calculates the counts of unique airline names in the 'airline' column of your DataFrame (`df`) and stores the counts in the `airline_counts` variable.

x = airline_counts.index: This line extracts the unique airline names as `x` (x-values) from the index of the `airline_counts` Series.

y = airline_counts.values: This line extracts the corresponding counts of each unique airline as `y` (y-values) from the values of the `airline_counts` Series.

plt.figure(figsize=(10, 6)): This code sets the figure size of the plot to 10 inches in width and 6 inches in height, determining the dimensions of the plot.

plt.bar(x, y, color='#de122a'): This line creates a bar plot with `x` as the x-values (airline names) and `y` as the y-values (counts). The bars are colored with the custom color '#de122a'.

plt.xlabel('Negative Reason', color='green'): This sets the label for the x-axis to 'Negative Reason' and customizes the label text color to 'green'.

plt.ylabel('Count', color='green'): This sets the label for the y-axis to 'Count' and customizes the label text color to 'green'.

plt.title('Sentiment Distribution by Negative Reason', color='blue'): This sets the title of the plot to 'Sentiment Distribution by Negative Reason' and customizes the title text color to 'blue'.

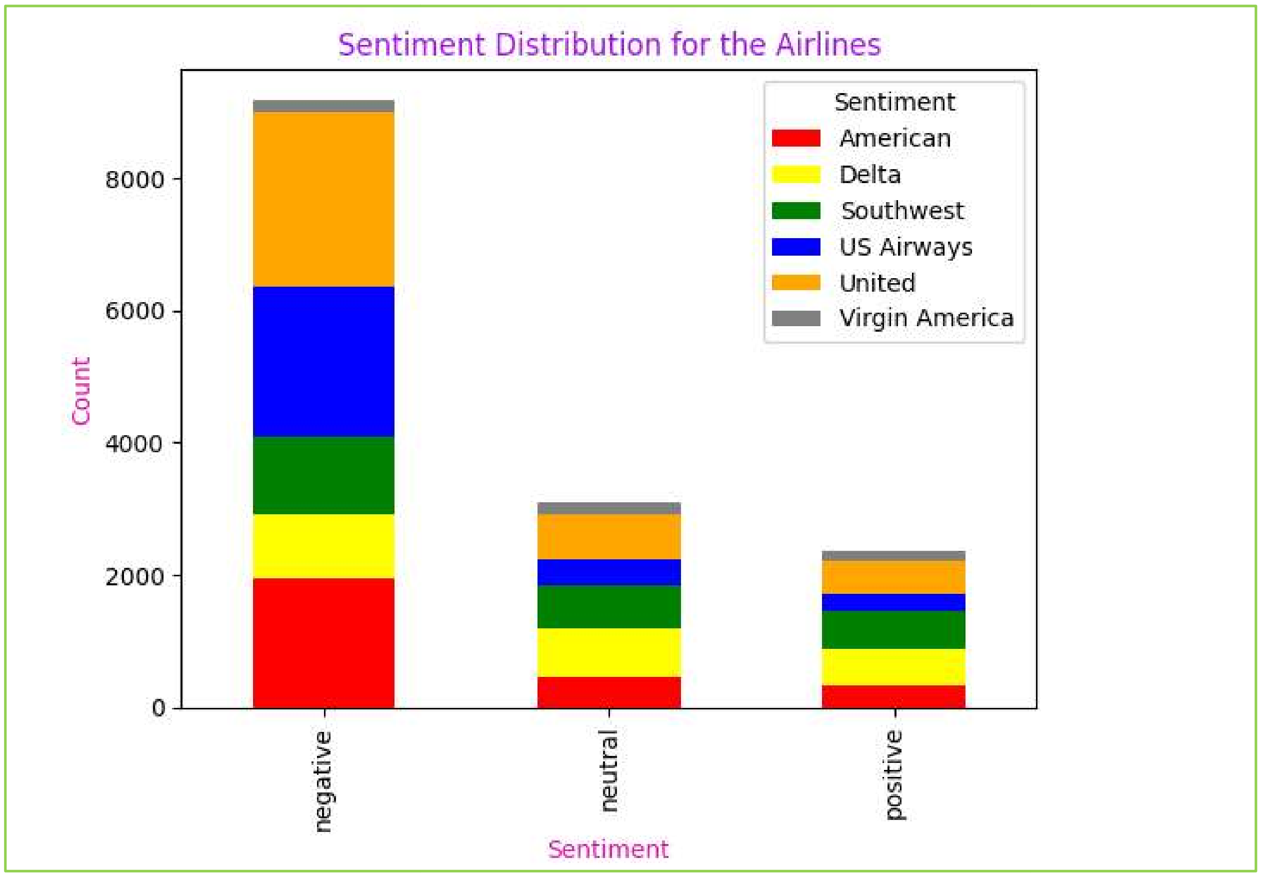
plt.xticks(rotation=90): This line rotates the x-axis labels (airline names) by 90 degrees to make them more readable when dealing with long labels.

plt.show(): This line displays the bar plot in your Python environment.

CODE:

```
sentiment_counts = df.groupby(['airline_sentiment', 'airline']).size().unstack(fill_value=0)
colors = ['red', 'yellow', 'green','blue','orange','grey']
sentiment_counts.plot(kind='bar', stacked=True, color=colors)
plt.xlabel('Airline',color='#e310ab')
plt.ylabel('Count',color='#e310ab')
plt.title('Sentiment Distribution by Airline',color='#a114de')
plt.legend(title='Sentiment', loc='upper right')
plt.show()
```

OUTPUT:



EXPLANATION:

sentiment_counts = df.groupby(['airline_sentiment', 'airline']).size().unstack(fill_value=0): This code groups the data in your DataFrame (`df`) by two columns: 'airline_sentiment' and 'airline'. It then computes the size (count) of each sentiment category within each airline group. The `unstack` operation pivots the data to create a table where each airline is a column, and the sentiment categories are rows. The `fill_value=0` parameter fills missing values with zeros.

colors = ['red', 'yellow', 'green', 'blue', 'orange', 'grey']: This list defines a set of colors for each sentiment category or airline. These colors will be used in the stacked bar plot.

sentiment_counts.plot(kind='bar', stacked=True, color=colors): This line creates a stacked bar plot using the `plot` method from the DataFrame `sentiment_counts`. The `kind='bar'` parameter specifies the type of plot, and `stacked=True` stacks the bars for each airline on top of each other, showing the sentiment distribution.

plt.xlabel('Airline', color='#e310ab'): This sets the label for the x-axis to 'Airline' and customizes the label text color to '#e310ab'.

plt.ylabel('Count', color='#e310ab'): This sets the label for the y-axis to 'Count' and customizes the label text color to '#e310ab'.

plt.title('Sentiment Distribution by Airline', color='#a114de'): This sets the title of the plot to 'Sentiment Distribution by Airline' and customizes the title text color to '#a114de'.

plt.legend(title='Sentiment', loc='upper right'): This line adds a legend to the plot. The `title` parameter sets the title of the legend to 'Sentiment', and the `loc` parameter specifies the legend's position, which is 'upper right' in this case.

plt.show(): This line displays the stacked bar plot in your Python environment.

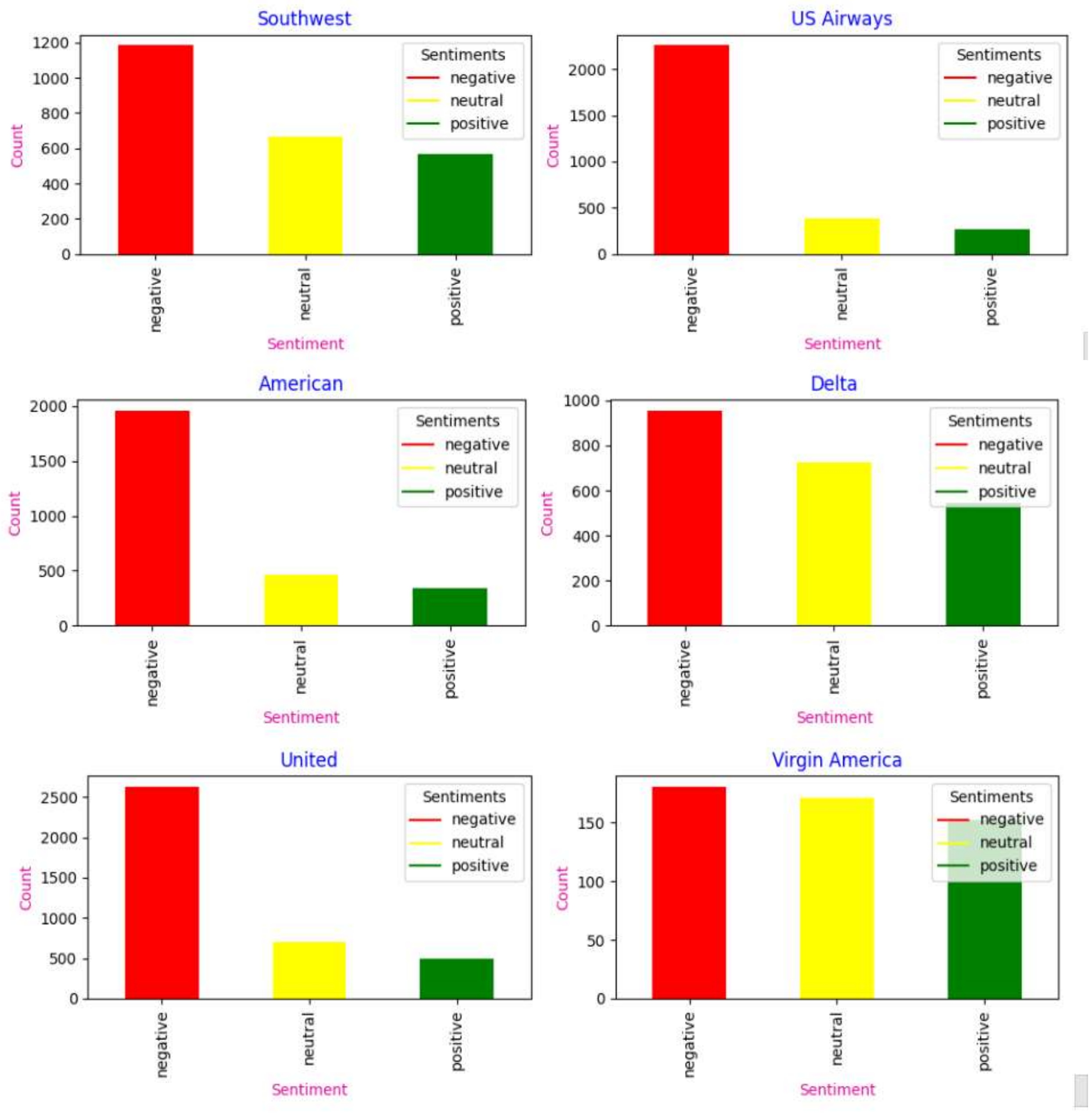
CODE:

```
sentiment_counts = df.groupby(['airline', 'airline_sentiment']).size().unstack(fill_value=0)
unique_airlines = sentiment_counts.index
fig, axes = plt.subplots(3, 2, figsize=(10, 10))
axes = axes.flatten()
colors = ['red', 'yellow', 'green']
legend_dict = {
    'negative': 'red',
    'neutral': 'yellow',
    'positive': 'green'
}

for i, j in enumerate(unique_airlines):
    sentiment_counts.loc[j].plot(kind='bar', stacked=True, ax=axes[i], color=[legend_dict[c] for c in
sentiment_counts.columns])
    axes[i].set_title(j,color='blue')
    axes[i].set_xlabel('Sentiment',color='#e6079b')
    axes[i].set_ylabel('Count',color='#e6079b')
    legend_handles = [plt.Line2D([0], [0], color=legend_dict[sentiment], label=sentiment) for sentiment in
sentiment_counts.columns]
    axes[i].legend(handles=legend_handles, title='Sentiments', loc='upper right')

plt.tight_layout()
plt.show()
```

OUTPUT:



EXPLANATION:

sentiment_counts = df.groupby(['airline', 'airline_sentiment']).size().unstack(fill_value=0): This line groups the data in your DataFrame (`df`) by two columns: 'airline' and 'airline_sentiment'. It then computes the size (count) of each sentiment category within each airline group. The data is unstacked to create a table where each airline is a row, and the sentiment categories are columns.

fig, axes = plt.subplots(3, 2, figsize=(10, 10)): This line creates a grid of subplots with 3 rows and 2 columns, resulting in a total of 6 subplots. The `fig` variable represents the figure, and `axes` is an array of subplot axes.

colors = ['red', 'yellow', 'green']: This list defines colors for the sentiment categories ('negative', 'neutral', and 'positive').

legend_dict: This dictionary maps sentiment categories to colors, allowing you to use the specified colors in the legend.

for i, j in enumerate(unique_airlines): This loop iterates through the unique airlines in your dataset.

sentiment_counts.loc[j].plot(kind='bar', stacked=True, ax=axes[i], color=[legend_dict[c] for c in sentiment_counts.columns]): For each airline, it creates a stacked bar plot using the counts of sentiment categories. The `stacked=True` parameter stacks the bars, and the colors are customized based on the legend dictionary.

axes[i].set_title(j, color='blue'): It sets the title of each subplot to the airline name, with the title text color in blue.

axes[i].set_xlabel('Sentiment', color='#e6079b'): It sets the x-axis label to 'Sentiment' with the label text color in a custom shade of pink.

axes[i].set_ylabel('Count', color='#e6079b'): It sets the y-axis label to 'Count' with the label text color in the same custom pink shade.

A custom legend is created for each subplot using `legend_handles` to specify the legend labels and colors. The legend is positioned in the upper right corner of each subplot.

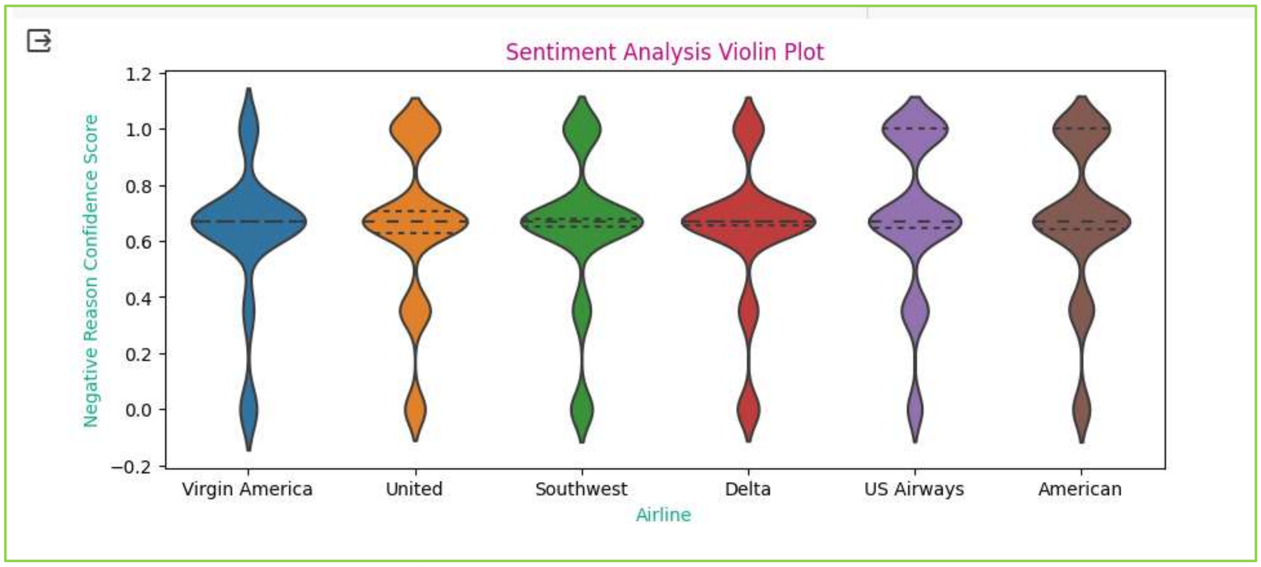
plt.tight_layout(): This function optimizes the subplot layout to prevent overlapping.

plt.show(): This line displays the entire plot, which consists of the stacked bar plots for each airline.

CODE:

```
plt.figure(figsize=(10, 6))
sns.violinplot(x='airline', y='negativereason_confidence', data=df, inner='quartile')
plt.xlabel('Airline',color='#0ca889')
plt.ylabel('Negative Reason Confidence Score',color='#0ca889')
plt.title('Sentiment Analysis Violin Plot',color='#bd0981')
plt.show()
```

OUTPUT:



EXPLANATION:

plt.figure(figsize=(10, 6)): This line sets the figure size of the plot to 10 inches in width and 6 inches in height. It determines the dimensions of the plot.

sns.violinplot(x='airline', y='negativereason_confidence', data=df, inner='quartile'): This line creates the violin plot using `seaborn`. Here's what each part of the code does:

x='airline': This specifies the data to be plotted on the x-axis, which is the 'airline' column in your DataFrame (`df`).

y='negativereason_confidence': This specifies the data to be plotted on the y-axis, which is the 'negativereason_confidence' column in your DataFrame.

data=df: You specify the DataFrame that contains the data to be plotted.

inner='quartile': This parameter configures the inner part of the violins to show quartiles, providing information about the distribution of data within each category.

plt.xlabel('Airline', color='#0ca889'): This sets the label for the x-axis as 'Airline' and customizes the label text color to '#0ca889'.

plt.ylabel('Negative Reason Confidence Score', color='#0ca889'): This sets the label for the y-axis as 'Negative Reason Confidence Score' and customizes the label text color to '#0ca889'.

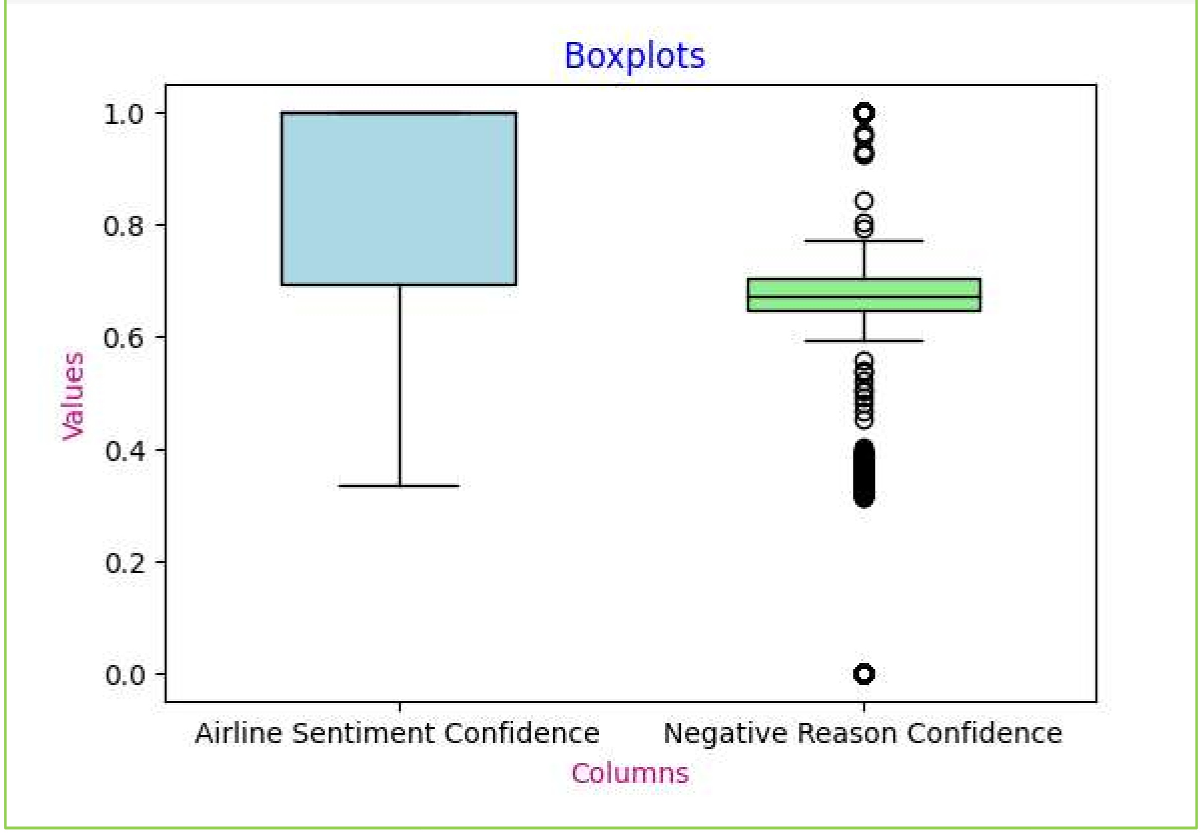
plt.title('Sentiment Analysis Violin Plot', color='#bd0981'): This sets the title of the plot to 'Sentiment Analysis Violin Plot' and customizes the title text color to '#bd0981'.

plt.show(): This line displays the violin plot in your Python environment.

CODE:

```
data1 = df['airline_sentiment_confidence']
data2 = df['negativereason_confidence']
plt.figure(figsize=(6, 4))
bp1 = plt.boxplot(data1, positions=[1], patch_artist=True, widths=0.5)
bp2 = plt.boxplot(data2, positions=[2], patch_artist=True, widths=0.5)
box_colors = ['lightblue', 'lightgreen']
whisker_color = 'black'
for bplot, color in zip([bp1, bp2], box_colors):
    for element in ['boxes', 'whiskers', 'medians', 'fliers']:
        plt.setp(bplot[element], color=whisker_color)
        if element == 'boxes':
            plt.setp(bplot[element], facecolor=color)
plt.xticks([1, 2], ['Airline Sentiment Confidence', 'Negative Reason Confidence'])
plt.xlabel('Columns',color='#bd0981')
plt.ylabel('Values',color='#bd0981')
plt.title(' Boxplots',color='blue')
plt.show()
```

OUTPUT:



EXPLANATION:

Two columns from the DataFrame **`df`** are extracted and assigned to **`data1`** and **`data2`** variables. These columns are **'airline_sentiment_confidence'** and **'negativereason_confidence'**, respectively.

plt.figure(figsize=(6, 4)):A figure is created with a specified size of 6x4 inches.

plt.boxplot():To create a box plot, The **`positions`** parameter is used to specify the positions of the boxplots on the x-axis. In this case, the **'airline_sentiment_confidence'** boxplot is positioned at 1, and the **'negativereason_confidence'** boxplot is positioned at 2. The **`patch_artist=True`** argument is used to fill the box with color, and `widths=0.5` sets the width of the boxes.

Two colors are defined for the boxplots. **'lightblue'** is used for **the 'airline_sentiment_confidence'** boxplot, and **'lightgreen'** is used for the **'negativereason_confidence'** boxplot. These colors are defined in the `box_colors` list.

A loop iterates through the two boxplots (bp1 and bp2) and the four boxplot elements (boxes, whiskers, medians, and fliers). It sets the colors for these elements. The boxes and whiskers are set to the specified colors in **`box_colors`,** while medians and fliers are set to 'black'.

The **`plt.xticks`** function is used to label the x-axis with the column names. The positions are set as [1, 2], corresponding to the positions of the boxplots. The labels are 'Airline Sentiment Confidence' and 'Negative Reason Confidence'.

The x-axis and y-axis labels are customized using **`plt.xlabel`** and **`plt.ylabel`.** In this case, they are given the text color '#bd0981' (a shade of pink).

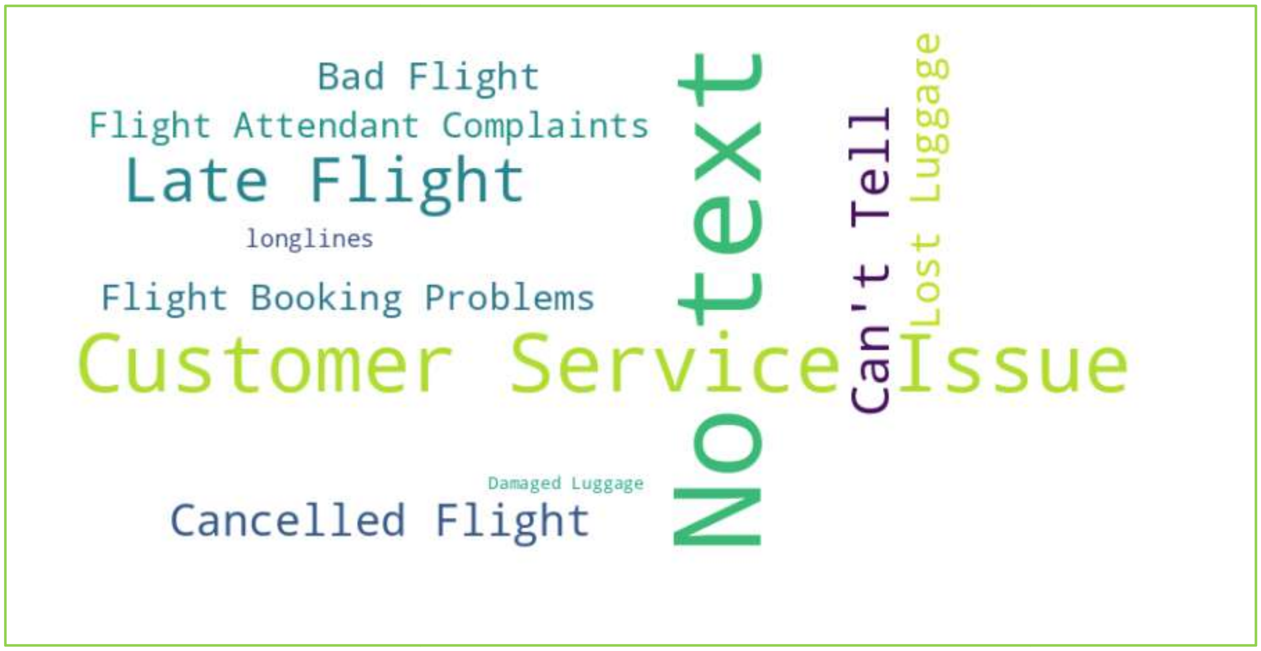
Plt.title() set the title as 'Boxplots' with the text color 'blue'.

plt.show() function is called to display the boxplot.

CODE:

```
from collections import Counter
word_counts = Counter(df['negativereason'])
from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate_from_frequencies(word_counts)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

OUTPUT:



EXPLANATION:

Import the `Counter` class from the `collections` module to count word frequencies.

Import the `WordCloud` class from the `wordcloud` library to create the word cloud visualization.

wordcloud = WordCloud(width=800, height=400, background_color='white') .generate _from_frequencies(word_counts): Use the `WordCloud` class to generate a word cloud visualization.

width and height: These parameters determine the dimensions of the word cloud image.

background_color: You set the background color to 'white' to make the word cloud's background white.

generate_from_frequencies: You generate the word cloud from the word frequency counts obtained using the `Counter` class.

plt.figure(figsize=(10, 5)): Create a Matplotlib figure with a specified figure size (width and height).

plt.imshow(wordcloud, interpolation='bilinear'):. Display the word cloud image.The `interpolation` parameter controls how the word cloud is rendered and can affect the image's smoothness.

plt.axis('off') to hide the axis labels and ticks in the plot.

plt.show() to display the word cloud image in your Python environment.