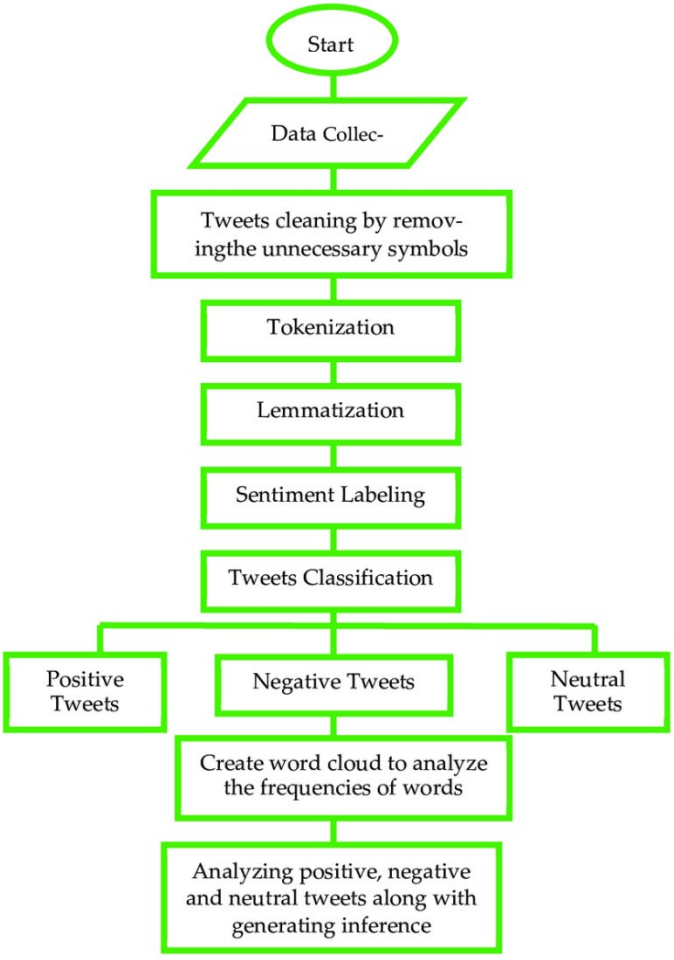
SENTIMENTAL ANALAYSIS FOR MARKETING

Data Preprocessing:

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the data mining process. More recently, data preprocessing techniques have been adapted for training machine learning models and AI models and for running inferences against them.

Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate results



.

Importing Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

- **import pandas as pd**: This line imports the pandas library, which is commonly used for data manipulation and analysis. The alias 'pd' is a convention for easier and more concise use of the library.
- **import numpy as np:** This line imports the NumPy library, which provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays. The alias 'np' is a common convention.
- **import matplotlib.pyplot as plt:** This line imports the pyplot module from the Matplotlib library. Matplotlib is a popular plotting library, and pyplot provides a convenient interface for **creating various types of plots.**
- **import seaborn as sns:** This line imports the seaborn library, which is built on top of Matplotlib and provides a high-level interface for creating informative and attractive statistical graphics.
- **from sklearn.model_selection import train_test_split:** This line imports the train_test_split function from scikit-learn. This function is used to split datasets into training and testing sets, which is crucial in machine learning for evaluating model performance.

Ignore Warnings:

```
import warnings
warnings.filterwarnings('ignore')
```

- **warnings.filterwarnings('ignore'):** This line configures the code to ignore any warnings that might be issued during its execution. While warnings can be informative, they are sometimes suppressed to enhance the clarity of the output.

Display Options for Pandas:

```
pd.set_option('display.max_columns',None)
```

- **pd.set_option('display.max_columns', None):** This line sets a pandas display option to show all columns when displaying a DataFrame. This is helpful when working with

datasets that have a large number of columns, ensuring that you can see all the information in the DataFrame.

Reading a CSV File into a DataFrame:

```
df = pd.read_csv("Tweets.csv")
```

- **pd.read_csv("Tweets.csv"):** This line uses the read_csv function from the pandas library to read a CSV (Comma-Separated Values) file named "Tweets.csv" into a pandas DataFrame. The DataFrame is a two-dimensional tabular data structure, and it's a primary data structure used in pandas for data manipulation and analysis.

Displaying the First Few Rows of the DataFrame:

```
df.head()
```

- **df.head():** After reading the CSV file and storing the data in the DataFrame variable df, this line uses the head() method to display the first few rows of the DataFrame. By default, head() shows the first 5 rows, providing a quick overview of the structure and content of the dataset.

Data visualization:

Function Definition (plot_sentiment):

```
def plot_sentiment(Airline):
    df1 = df[df['airline'] == Airline]
    count = df1['airline_sentiment'].value_counts().reset_index().rename(columns={'index': Airline, 'airline_sentiment': 'count'})
    sns.barplot(data=count, x=Airline, y='count')
```

- This function takes an airline name (**Airline**) as an argument.
- It filters the DataFrame df to create a new DataFrame (**df1**) containing only rows where the airline column is equal to the specified airline.
- It then counts the occurrences of each sentiment category in the 'airline_sentiment' column and stores the result in the count DataFrame.
- Finally, it creates a bar plot using Seaborn (**sns.barplot**) where the x-axis represents the specified airline (**Airline**), and the y-axis represents the count of each sentiment.

Figure and Subplots:

```
plt.figure(figsize=(15, 7))
plt.subplot(2, 3, 1)
plot_sentiment('United')
plt.subplot(2, 3, 2)
plot_sentiment('US Airways')
plt.subplot(2, 3, 3)
plot_sentiment('American')
plt.subplot(2, 3, 4)
plot_sentiment('Southwest')
plt.subplot(2, 3, 5)
plot_sentiment('Delta')
plt.subplot(2, 3, 6)
plot_sentiment('Virgin America')
```

- It creates a new figure with a size of 15x7 inches (**plt.figure(figsize=(15, 7))**).
- It defines a 2x3 subplot grid and specifies the position for each subplot using **plt.subplot(2, 3, i)** where **i** ranges from 1 to 6.
- For each subplot, it calls the **plot_sentiment** function with a specific airline as an argument, generating a bar plot for the sentiment distribution of that airline.

Layout and Display:

- **plt.tight_layout()** adjusts the spacing between subplots for better layout.
- **plt.show()** displays the final plot.

Conclusion:

In short, we've just prepared your data for sentiment analysis! We used some handy tools to organize everything. First, we put your info from a CSV file into a special table. Then, we made the text neat and clean, removing extra stuff. We turned sentiments, like happy or sad, into numbers so computers can understand them. Lastly, we split the data into parts for training and testing our analysis. Now, your data is all set for making a cool sentiment analysis model. Have fun- exploring and learning!