SENTIMENT ANALYSIS FOR MARKETING

Problem Definition :

To start building a sentiment analysis solution, we need to load and preprocess the dataset. In this case, we'll use the Twitter Airline Sentiment dataset.

1. Load the Dataset:

Download the dataset from Kaggle and save it to a local directory. Then, you can load the dataset using a library like Pandas.

```
import pandas as pd
# Load the dataset
df = pd.read_csv('Tweets.csv')
```

2. Preprocess the Data:

Data preprocessing is a crucial step in sentiment analysis. Common preprocessing steps include:

Removing unnecessary columns: Depending on your analysis, you may not need all the columns in the dataset.
Cleaning text data: This involves removing HTML tags, special characters, and irrelevant symbols.
Tokenization: Splitting text into words or tokens.
Lowercasing: Ensuring uniformity by converting text to lowercase.
Removing stopwords: Eliminating common words that don't carry significant meaning

```
import re
import nltk
from nltk.corpus import stopwords
```

```
# Download NLTK data
nltk.download('stopwords')

# Remove unnecessary columns
df = df[['text', 'airline_sentiment']]

# Clean and preprocess text data
def preprocess_text(text):
    # Remove HTML tags
    text = re.sub(r'<.*?>', '', text)
    # Remove special characters and symbols
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Tokenization and lowercasing
    words = text.lower().split()
    # Removing stopwords
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)

df['text'] = df['text'].apply(preprocess_text)
```

3. Exploring the Preprocessed Data:

After preprocessing, it's beneficial to explore the data to ensure it has been cleaned properly. You can check the first few rows of the preprocessed data

```
# Display the first few rows of the preprocessed data
print(df['text'].head())
```

5. Label Encoding:

In sentiment analysis, labels are often converted into numerical values for model training. You can encode sentiment labels (e.g., 'positive,' 'negative,' 'neutral') into numerical values (e.g., 0, 1, 2) using label encoding

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['sentiment_encoded'] =
label_encoder.fit_transform(df['airline_sentiment'])
```

6. Splitting the Data:

Split the dataset into training and testing sets, which is essential for model evaluation

```
from sklearn.model_selection import train_test_split

X = df['text']
y = df['sentiment_encoded']

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

7. Vectorization:

To use machine learning or deep learning models, you'll need to vectorize the text data. Common approaches include TF-IDF vectorization or word embeddings

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

8. conclusion:

sentiment analysis solution is builded successfully and Twitter Airline Sentiment dataset is loaded and preprocessed successfully.