Sentiment Analysis For Marketing

Phase 4 :

Development Part 2 :

Developing the sentiment analysis solution by:

- Employing NLP techniques.
- Generating insights.

Phase 4 represents a critical stage in the project's development journey. In this phase, we continue to enhance the Sentiment Analysis for Marketing by integrating it with a NLP techniques. Through comprehensive analysis, we intend to uncover trends, sentiments, and customer opinions that can provide critical intelligence for the airline industry.

Data Source:

Link provide:

https://www.kaggle.com/datasets/crowdflower/twitterairlinesentiment

Necessary step to follow:

Certainly, here are the eight steps explained briefly for the given code:

1. Import Libraries: The first step is to import the necessary libraries using the 'import' statements. In this code, libraries like `pandas`, `re`, `nltk`, `matplotlib`, and `seaborn` are imported to handle data, text processing, and visualization.

2. Load Data: The code loads a CSV file (`'Tweets.csv'`) using the

‘pd.read_csv()’ function from the Pandas library. This data is stored in the ‘airline_tweets’ DataFrame.

3. Adjust Plot Size: The code modifies the default plot size using the `plt.rcParams["figure.figsize"]` to make plots larger. It prints the original plot size and sets it to a new size of 8x6.

4. Create Pie Charts: The code uses Pandas to create two pie charts displaying the distribution of sentiment and airline types. It sets the chart type as 'pie' and adds percentage labels using `autopct`.

5. Create a Bar Plot: It groups the data in the `airline_tweets` DataFrame by the columns 'airline' and 'airline_sentiment' and counts the number of occurrences. Then, it plots a bar chart to visualize the sentiment distribution for each airline.

6. Text Preprocessing: This step preprocesses the text data in the 'airline_tweets' DataFrame. It removes special characters, single characters, multiple spaces, 'b' prefixes, and converts the text to lowercase. The processed text is stored in the `processed_features` list.

7. Text Vectorization: The code uses the `TfidfVectorizer` from scikit-learn to convert the processed text data into numerical features. It creates a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) vectors with a specified maximum number of features, minimum document frequency, and maximum document frequency.

8. Machine Learning Model: This step involves preparing the data for training and testing. It splits the data into training and testing sets using

‘train_test_split’. Then, it creates a Random Forest Classifier, fits it to the training data, makes predictions on the test data, and evaluates the model's performance using a confusion matrix and accuracy score.

These steps provide a high-level overview of what the code does, from data loading and visualization to text preprocessing and machine learning model training and evaluation.

Python code for Sentiment analysis for Marketing using TF-IDF:

```
# Step 1: Import Libraries
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
%matplotlib inline

# Step 2: Load Data
airline_tweets = pd.read_csv(r'D:\Tweets.csv')

# Step 3: Adjust Plot Size
plot_size = plt.rcParams["figure.figsize"]
print(plot_size[0])
print(plot_size[1])
plot_size[0] = 8
plot_size[1] = 6
plt.rcParams["figure.figsize"] = plot_size

# Step 4: Create Pie Charts
 airline_tweets.airline.value_counts().plot(kind='pie', autopct='%1.0f%%')
airline_tweets.airline_sentiment.value_counts().plot(kind='pie',
autopct='%1.0f%%', colors=["brown", "gold", "blue"])
```

```
# Step 5: Create a Bar Plot
airline_sentiment = airline_tweets.groupby(['airline',
'airline_sentiment']).airline_sentiment.count().unstack()
airline_sentiment.plot(kind='bar')
import seaborn as sns
sns.barplot(x=’airline_sentiment’ , y=’airline_sentiment_confidence’ ,
data=airline_tweets)


# Step 6: Text Preprocessing
features = airline_tweets.iloc[:, 10].values
labels = airline_tweets.iloc[:, 1].values
processed_features = []
for sentence in range(0, len(features)):
 # Remove special characters, single characters, multiple spaces, 'b' prefixes, and
convert to lowercase
 processed_feature = re.sub(r'\W', ' ', str(features[sentence]))
 processed_feature = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)
processed_feature = re.sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)
processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)
processed_feature = re.sub(r'^b\s+', '', processed_feature)
processed_feature = processed_feature.lower()
processed_features.append(processed_feature)


# Step 7: Text Vectorization
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=2500, min_df=7, max_df=0.8,
stop_words=stopwords.words('english'))
processed_features = vectorizer.fit_transform(processed_features).toarray()


# Step 8: Machine Learning Model
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(processed_features, labels,
test_size=0.2, random_state=0)
from sklearn.ensemble import RandomForestClassifier
```

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)

predictions = text_classifier.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test, predictions))

print('accuracy score', accuracy_score(y_test, predictions))

OUTPUT:

```
In [21]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(processed_features, labels, test_size=0.2, random_state=0)

In [22]: from sklearn.ensemble import RandomForestClassifier

         text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
         text_classifier.fit(X_train, y_train)

Out[22]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=200,
                                n_jobs=None, oob_score=False, random_state=0, verbose=0,
                                warm_start=False)

In [23]: predictions = text_classifier.predict(X_test)

In [24]: from sklearn.metrics import confusion_matrix, accuracy_score

         print(confusion_matrix(y_test,predictions))
         print('accuracy score',accuracy_score(y_test, predictions))

[[1723  108   39]
 [ 326  248   40]
 [ 132   58  254]]
accuracy score 0.7599043715846995
```

Conclusion:

In conclusion, the provided code snippet demonstrates a data analysis and machine learning pipeline for sentiment analysis of airline-related tweets. It begins by importing necessary libraries, loads the dataset, and customizes plot sizes for data visualization. The code visualizes sentiment and airline distributions with pie charts and sentiment breakdowns for each airline with a bar chart.

It then proceeds to text preprocessing, where it cleans and prepares the text data for analysis. Afterward, it employs TF-IDF vectorization to convert the text into numerical features for machine learning. Finally, it trains a Random Forest Classifier to predict sentiment labels and evaluates the model's performance using a confusion matrix and accuracy score.

This code serves as a comprehensive example of how to perform sentiment analysis on textual data, showcasing data preprocessing, feature engineering, and machine learning model training. It can be a useful starting point for sentiment analysis tasks in the domain of airline-related tweets.