Sentiment analysis for marketing

Data Preprocessing Techniques

Preprocessing Steps

Data cleaning:

- Remove Duplicates
- Identify Missing Values
- Remove Irrelevant Columns

Text Preprocessing:

- Tokenization
- Lowercasing
- Stopword Removal
- Lemmatization

Data Exploration:

- Summary Statistics
- Data Visualization

Remove Duplicates

Identify Duplicates: Use Python libraries like Pandas to identify duplicate rows in your dataset.

```
[ ]  # Import necessary libraries
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import nltk
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize
     from nltk.stem import WordNetLemmatizer
     from sklearn.feature_extraction.text import TfidfVectorizer

[ ]  # Load the dataset
     data = pd.read_csv("/content/Tweets.csv")

[ ]  # Data Cleaning
     # Remove duplicates
     data = data.drop_duplicates()
```

Identify Missing Values

Delete Rows/Columns: In some cases, you may choose to remove rows or columns with excessive missing data.

```
[ ]  # Handle missing values
     data = data.dropna()
```

Remove Irrelevant Columns:

Irrelevant columns do not contribute to your analysis or modeling. To deal with them:

- **Identify Irrelevant Columns:** Review the dataset and domain knowledge to determine which columns are irrelevant.
- **Drop Columns:** Use Pandas to remove these columns from the dataset.

```
[ ]  # Remove irrelevant columns
     data = data[['airline_sentiment', 'text']]
```

Tokenization:

Tokenization is the process of breaking text into individual words or tokens. This makes it easier to analyze and work with text data.

- **Tokenization in Python:**

You can use libraries like NLTK or spaCy for tokenization.

```
     # Text Preprocessing
     # Tokenization
     nltk.download('punkt')
     data['text'] = data['text'].apply(word_tokenize)
```

Lowercasing:

Lowercasing involves converting all text to lowercase. This ensures that words are treated consistently, regardless of their original casing.

- **Lowercasing in Python:**

You can use the **.lower()** method to convert text to lowercase.

```
[ ]  # Lowercasing
     data['text'] = data['text'].apply(lambda x: [word.lower() for word in x])
```

Stopword Removal:

Stopwords are common words (e.g., "the," "and," "in") that don't typically carry significant meaning in text analysis. Removing them can reduce noise in the data.

- **Stopword Removal in Python:**
  You can use libraries like NLTK or spaCy to remove stopwords.

```
[ ]  # Stopword Removal
     nltk.download('stopwords')
     stop_words = set(stopwords.words('english'))
     data['text'] = data['text'].apply(lambda x: [word for word in x if word not in stop_words])
```

Lemmatization:

Lemmatization and stemming both aim to reduce words to their base or root form, making different inflections or forms of a word equivalent. Lemmatization is more precise but can be slower than stemming.

- **Lemmatization in Python:**

You can use NLTK or spaCy for lemmatization.

```
     # Lemmatization
     nltk.download('wordnet')
     lemmatizer = WordNetLemmatizer()
     data['text'] = data['text'].apply(lambda x: [lemmatizer.lemmatize(word) for word in x])
```

Summary Statistics:

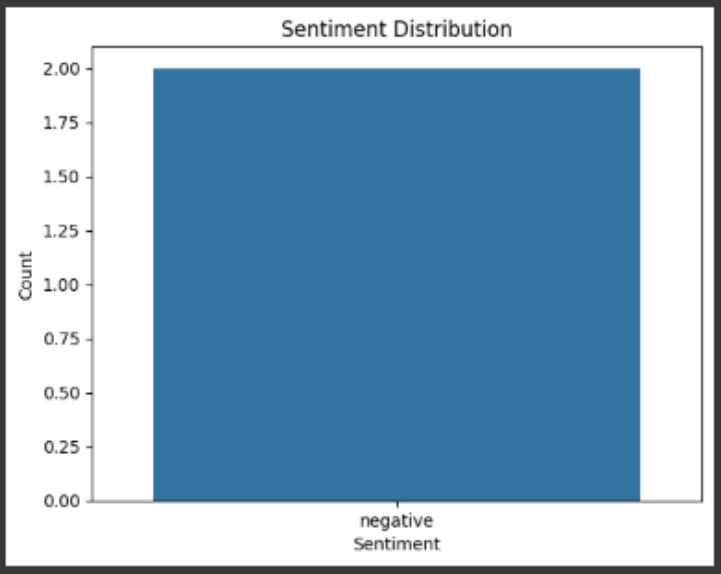
Summary statistics provide an overview of the central tendencies and variability of the dataset. You can use Python libraries like Pandas to compute these statistics. Here's a sample table showcasing summary statistics for a dataset:

```
[ ]  # Data Exploration
     # Summary statistics
     sentiment_counts = data['airline_sentiment'].value_counts()
```

Data Visualization:

Data visualizations are essential for a deeper understanding of the data distribution. You can use libraries like Matplotlib, Seaborn, or Plotly for creating various types of plots. Here's an example of code to create common visualizations:

```python
# Plot sentiment distribution
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Distribution')
plt.show()
```




```python
[ ] # TF-IDF Vectorization
    tfidf_vectorizer = TfidfVectorizer(max_features=1000)
    tfidf_matrix = tfidf_vectorizer.fit_transform(data['text'].apply(lambda x: ' '.join(x)))
```