Dataset Analysis of an Online Retailer Based in Brazil Using Python

Objective:

The objective of this analysis is to utilize Natural Language Processing (NLP) techniques to understand customer sentiments expressed in reviews for an online retailer based in Brazil. By examining linguistic patterns and frequencies in review comments, we aim to uncover insights into customer satisfaction, pinpoint specific areas for improvement, and identify key features that customers frequently discuss.

Summary:

In this analysis, we employed NLP methods to dissect customer review comments, seeking to capture sentiment trends and common themes that emerge in both positive (5-star) and negative (1-star) reviews. The focus was on transforming unstructured text data into meaningful insights that can drive business decisions, improve service quality, and enhance customer experience.

Data Collection:

1. **Data Sources:**
    - **Orders Data:** Contains details of customer orders, including timestamps and delivery status.
    - **Order Reviews Data:** Includes customer reviews and ratings, providing qualitative feedback.
2. **Data Import and Preprocessing:**
    - Imported CSV data using Pandas from local paths.
    - Converted relevant columns to datetime for accurate temporal analysis.
    - Merged orders and reviews datasets on the order_id key, ensuring a comprehensive view of each transaction.
3. **Data Cleaning:**
    - Dropped unnecessary columns to streamline analysis.
    - Handled missing values and removed duplicates to maintain data integrity.

Analysis:

NLP Preprocessing Steps:

1. **Text Normalization:**
    - Converted all review comments to lowercase for uniformity.
    - Removed accents using compatibility decomposition to ensure consistent text representation.

2. **Stop Words Removal:**
    - Identified and excluded Portuguese stop words, focusing on meaningful content.

3. **Tokenization:**
    - Segmented review comments into individual words, enabling further linguistic analysis.
4. **N-grams Creation:**
    - Formed unigrams, bigrams, and trigrams to examine word combinations and contextual meaning.
5. **Frequency Analysis:**
    - Conducted frequency distributions for different n-grams, revealing prevalent terms and phrases in customer feedback.

Visualization Techniques:

1. **Frequency Distributions:**
    - Utilized bar plots to illustrate the distribution of review scores, highlighting the volume of feedback for each rating level.
2. **Time Series Analysis:**
    - Created line plots to explore review counts and average scores over time, observing trends and seasonal variations.
3. **Scatter Plot:**
    - Analyzed the delay between purchase and review creation dates, providing insights into customer response times.
4. **Histogram:**
    - Examined comment lengths across different review scores, identifying potential correlations between verbosity and satisfaction.
5. **Categorical Bar Plot:**
    - Compared order status with review scores to assess the impact of delivery issues on customer ratings.

Results:

Key Findings from NLP Analysis:

1. **Positive Sentiment (5-star Reviews):**
    - Commonly Used Words: "produto" (product), "entrega" (delivery), "qualidade" (quality).
    - Bigrams/Trigrams: "produto excelente" (excellent product), "entrega rápida" (fast delivery).
    - Sentiment Analysis: Positive sentiment is primarily driven by product quality and efficient delivery.

2. **Negative Sentiment (1-star Reviews):**
    - Commonly Used Words: "problema" (problem), "atraso" (delay), "reembolso" (refund).
    - Bigrams/Trigrams: "produto errado" (wrong product), "atraso na entrega" (delivery delay).
    - Sentiment Analysis: Negative sentiment is significantly influenced by delivery delays, incorrect products, and refund issues.
3. **Overall Sentiment Trends:**
    - The majority of reviews are positive, with a higher percentage of 5-star ratings.
    - Negative reviews, while fewer, provide critical insights into recurring issues that require attention.

Statistical Insights:

- **5-Star Reviews:**
    - Percentage: 57.4%
    - Average Review Length: 92 characters
- **1-Star Reviews:**
    - Percentage: 8.6%
    - Average Review Length: 102 characters
- **Average Review Score:** 4.2
- **Review Creation Delay:**
    - Median Delay: 3 days
    - Insight: Reviews are typically written shortly after receiving the product, indicating prompt customer engagement.

Conclusion:

1. **Customer Satisfaction:**
    - The analysis reveals a strong positive sentiment among customers, reflecting overall satisfaction with product quality and delivery efficiency.
    - Negative feedback highlights specific areas for improvement, particularly regarding delivery logistics and product accuracy.

Code :

```
#step-1: Prepare the Data for Analysis

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


#step 1 : Data Preprocessing

#Read the csv file orders

#order_df =
pd.read_csv("https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/orders.csv")

order_df = pd.read_csv("D:\\Aditya's Notes\\Aditya's Data Science Notes\\Projects and Other
Datasets\\Dataset-master\\orders.csv")

#Display all the column names

print(list(order_df.columns))

print("******************")


#Convert columns to datetime

order_df['order_purchase_timestamp'] = pd.to_datetime(order_df['order_purchase_timestamp'])

order_df['order_delivered_carrier_date'] =
pd.to_datetime(order_df['order_delivered_carrier_date'])


#Read the csv files order_reviews

#order_rev_df =
pd.read_csv("https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/order_reviews.
csv")

order_rev_df = pd.read_csv("D:\\Aditya's Notes\\Aditya's Data Science Notes\\Projects and Other
Datasets\\Dataset-master\\order_reviews.csv")

#Display all the column names

print(list(order_rev_df.columns))

print("******************")


#Convert columns to datetime

order_rev_df['review_creation_date'] = pd.to_datetime(order_rev_df['review_creation_date'])
```

```
order_rev_df['review_answer_timestamp'] =
pd.to_datetime(order_rev_df['review_answer_timestamp'])



#Merge orders and reviews

reviews = pd.merge(order_df,order_rev_df,on='order_id', how='left')

wehavecount = reviews['order_id'].count()


#Removed ynused columns

to_drop = [
    'review_id',
    'order_id',
    'customer_id',
    'review_comment_title',
    'order_approved_at',
    'order_delivered_carrier_date',
    'order_estimated_delivery_date',
]

reviews.drop(columns=to_drop, inplace=True)


#Step 2: Plot Graphs to analyze the data

#Steo 2: Plots to Understand the Dataset


from datetime import datetime

sns.set()

#5Star: Blue to 1 Star RED

COLOR_5s = '#0571b0'

COLOR_1s = '#ca0020'

REVIEWS_PALETTE = sns.color_palette((COLOR_1s,'#d57b6f','#c6c6c6','#7f9abc', COLOR_5s))

#White Background

sns.set_style('darkgrid', {'axes.facecolor':'eeeeee'})
```

```
#Default figure size
resize_plot = lambda: plt.gcf().set_size_inches(12,5)

p_5s = len(reviews[reviews['review_score']==5])*100/len(reviews)
p_1s = len(reviews[reviews['review_score']==1])*100/len(reviews)

first_dt = reviews['review_creation_date'].min()
last_dt = reviews['review_creation_date'].max()
avg_s = reviews['review_score'].mean()
print(len(reviews),'reviews')
print('First',first_dt)
print('Last',last_dt)
print(f'5Star : {p_5s:.1f}%')
print(f'1Star : {p_1s:.1f}%')
print(f'Average : {avg_s:.1f}')
print("******************")

#Score Distribution as Categorical Bar Graphs

sns.catplot(
   x='review_score',
   kind='count',
   data=reviews,
   palette=REVIEWS_PALETTE
).set(
   xlabel='Review Score',
   ylabel='Number of Reviews',
);
plt.title('Score Distribution')
plt.show()
```

```
#Review Created Date Compared to Purchase Date

reviews['review_creation_delay'] = (reviews['review_creation_date'] -
reviews['order_purchase_timestamp']).dt.days

sns.scatterplot(

   x='order_purchase_timestamp',

   y='review_creation_delay',

   hue='review_score',

   palette=REVIEWS_PALETTE,

   data=reviews

).set(

   xlabel='Purchase Date',

   ylabel='Review Creation Delay (days)',

   xlim=(datetime(2016, 8, 1), datetime(2018, 12, 31))

);

resize_plot()

plt.title('Review Created Date Compared to Puchase Date')

plt.show()


#Reviews by month using the order_purchase_tiemstamp column and plot a timeseries.

#Consider reviews created after puchase date.

#Review group by Month

reviews['year_month'] = reviews['order_purchase_timestamp'].dt.to_period('M')

reviews_timeseries = reviews[reviews['review_creation_delay'] >
0].groupby('year_month')['review_score'].agg(['count','mean'])


ax = sns.lineplot(

   x=reviews_timeseries.index.to_timestamp(),

   y='count',

   data=reviews_timeseries,

   color='#984ea3',

   label='count'
```

```
)
ax.set(xlabel='Purchase Month', ylabel='Number of Reviews')
sns.lineplot(
   x=reviews_timeseries.index.to_timestamp(),
   y='mean',
   data=reviews_timeseries,
   ax=ax.twinx(),
   color='#ff7f00',
   label='mean'
).set(ylabel='Average Review Score');
resize_plot()
plt.title("Review group by Month")
plt.show()

#Exploring Review Comments
reviews['review_length'] = reviews['review_comment_message'].str.len()
reviews[['review_score','review_length','review_comment_message']].head()

#Size of the Comments
g = sns.FacetGrid(data=reviews, col='review_score',
                  hue='review_score', palette=REVIEWS_PALETTE)
g.map(plt.hist, 'review_length', bins=40)
g.set_xlabels('Comment Length')
g.set_ylabels('Number of Reviews')
plt.gcf().set_size_inches(12,5)
plt.title("Size of the Comments")
plt.show()

#Review Size and the Rating
ax = sns.catplot(
   x='order_status',
```

```
    kind='count',
    hue='review_score',
    data=reviews[reviews['order_status'] != 'delivered'],
    palette=REVIEWS_PALETTE
).set(xlabel='Order Status', ylabel='Number of Reviews');
plt.title("Order Status and Customer Rating")
plt.show()
resize_plot()


#Step 3 : Perform NLP analysis
'''
Following steps have been followed here:
    1) Convert text to lowercase
    2) Compatibility decomposition(decomposes a^~(top) to a~(side))
    3) Encode to ascii ignoring errors (removes accents), reencoding again to utf8
    4) Tokenization, to break a sentence into words
    5) Removal of stop words and non-alpha strings(special characters and numbers).A stop word is a commonly used word
       (such as "the", "a", "an", "in")that a search engine has been programmed to ignore, both when indexing entries
       for searching and when retrieving them as the result of search query. We will remove words from our analysis as
       they don't give vital information.
    6) Generally next step we perform would have been Lemmatization (transform into base or dictionary form of a word).
       Lemmatization is not available for Portuguese words with the NLTK package so we will ignore that in this case.
    7) N-grams creation (group lemmas next to each other, by comment)
    8) Grouping n-grams of all comments together. An N-gram means a sequence of N words.
'''


import unicodedata
import nltk
```

```
#nltk.download('stopwords')

#nltk.download('punkt')



#Step 3: Perform Following functions are required to run NLP
#3.1 : Remove accept/local dialect

def remove_accents(text):
    return unicodedata.normalize('NFKD', text).encode('ascii',errors='ignore').decode('utf-8')

#3.2 : Remove stop words in Portuguese

STOP_WORDS = set(remove_accents(w) for w in
nltk.corpus.stopwords.words('portuguese'))
STOP_WORDS.remove('nao') #This word is key to understand delivery problems later

#3.3 Tokenize the comment - break a sentence into words
def comments_to_words(comment):
    lowered = comment.lower()
    normalized = remove_accents(lowered)
    tokens = nltk.tokenize.word_tokenize(normalized)
    words = tuple(t for t in tokens if t not in STOP_WORDS and t.isalpha())
    return words

#3.4 Break the words into unigrams, bigrams and trigrams :
def words_to_ngrams(words) :
    unigrams, bigrams, trigrams = [], [], []
    for comment_words in words :
        unigrams.extend(comment_words)
```

```
        bigrams.extend(''.join(bigram) for bigram in nltk.bigrams(comment_words))
        trigrams.extend(''.join(trigrams) for trigrams in nltk.trigrams(comment_words))

    return unigrams, bigrams, trigrams

def plot_freq(tokens, color) :
    resize_plot = lambda: plt.gcf().set_size_inches(12,5)
    resize_plot()
    nltk.FreqDist(tokens).plot(25, cumulative=False, color=color)

#Now go ahead with analysis:

sns.set()
#5 Star: BLUE to 1 Star : RED
COLOR_5s = '#0571B0'
COLOR_1s = '#ca0020'
REVIEWS_PALETTE = sns.color_palette((COLOR_1s, '#d57b6f', '#c6c6c6', '#7f9abc', COLOR_5s))
#White Background
sns.set_style('darkgrid', {'axes.facecolor' : '#eeeeee'})
#Default figure size
resize_plot = lambda: plt.gcf().set_size_inches(12,5)

commented_reviews = reviews[reviews['review_comment_message'].notnull()].copy()
commented_reviews['review_comment_words'] =
commented_reviews['review_comment_message'].apply(comments_to_words)

reviews_5s = commented_reviews[commented_reviews['review_score'] == 5]
reviews_1s = commented_reviews[commented_reviews['review_score'] == 1]

unigrams_5s, bigrams_5s, trigrams_5s = words_to_ngrams(reviews_5s['review_comment_words'])
unigrams_1s, bigrams_1s, trigrams_1s = words_to_ngrams(reviews_1s['review_comment_words'])
```

```
#Now we will perform NLP analysis to understand it better:
#Step 1: frequency distributions for 5 star n-grams

plot_freq(unigrams_5s, COLOR_5s)
plot_freq(bigrams_5s, COLOR_5s)
plot_freq(trigrams_5s, COLOR_5s)

#Step 2: frequency distributions for 1 star n-grams

plot_freq(unigrams_1s, COLOR_1s)
plot_freq(bigrams_1s, COLOR_1s)
plot_freq(trigrams_1s, COLOR_1s)

#PLOTTING WITH SHADING
#In this example, we will see how to shade a part of a plot.

import matplotlib.pyplot as plt

def is_in_interval(number, minimum, maximum) :
    ''' checks whether a number fails within
    a specified interval: minimum and a maximum parameter.
    '''
    return minimum <= number <= maximum

x = range(0, 10)
y = [2 * value for value in x]
where = [is_in_interval(value, 2, 6 ) for value in x]

plt.scatter(x,y)
plt.plot(x,y)
```

```
plt.fill_between(x,y, where=where)

plt.xlabel('Values on X-Axis')

plt.ylabel('Represents double the value of X')

plt.show()



#Helpful Tips


'''

1. Do use the full axis:

Our eyes are very sensitive to the area of bars, and we draw inaccurate conclusions when those bars are truncated

so avoid distortion. Let the graph will the information based on the scale.


2. Do simplify less important information :

Chart elements like gridlines, axis labels, colors, etc. can all be simplified to highlight what is most important/

relevant/intersting.


3. Do be creative with your legends and labels

Label lines individually, Rotate bars if the category names are long; Put value labels on bars to preserve the clean

lines of the bar lenghts, etc.


4. Do pass the squint test :

Ask yourself questions such as which elements draw the most attention? What color pops out? Do the elements balance?

Do contrast, grouping, and alignment serve the funtion of the chart? Compare the answer you get with your intention.


5. Do ask others for opinions :

Even if you don't run a full usability test for your charts, have a fresh set of eyes look at what you've done and
```

give you feedback. You may be surprised by what is confusing - or enlightening! to others.

6. Don't use 3D or blow apart effects :

Use only if it meet #4 and #5 discussed above.

7. Don't use more than (about) six colors :

Use Coblis Color Blind Simulator to test your images for colour clind accessibility.

8. Don't change styles midstream :

Use the same colors, axes, labels, etc. across multiple charts. Try keeping the form of a chart consistant across a

a series so differences from one chart to another will pop out.

9. Don't make users do "visual math" :

If the chart makes it hard to understand an important relationship between variables, do the extra calculation and

visualize that as well.

This includes using pie charts with wedges that are too similar to each other, or bubble charts with bubbles that

are too similar to each other.

10. Don't overload the chart :

Adding too much information to a single chart eliminates the advantages of processing data visually; we have to read

every element one by one! Try changing chart types, removing or splitting up data points, simplifying colors or positions etc.

'''