Sentiment Analysis of Product Review

Python Data Analysis Project

Introduction :

This project involves analyzing a dataset of customer reviews to understand sentiments expressed by users about products. Sentiment analysis is a key technique in natural language processing (NLP) and data analysis, used to extract subjective information from textual data. The goal is to uncover insights from the reviews and build models that can predict sentiments (positive, negative, or neutral).

Dataset Overview:

The provided dataset contains information about customer reviews for products in the "Books" category. Key columns include:

1. **Review Text**: Textual feedback from customers, the primary source for sentiment analysis.
2. **Sentiment**: Pre-labelled sentiment for each review (positive, negative, or neutral), useful for supervised machine learning.
3. **Star Rating**: Numeric rating (1–5 stars), which might correlate with sentiment.
4. **Votes (Helpful/Total)**: Indicators of review credibility and popularity.
5. **Verified Purchase**: Distinguishes between verified and non-verified reviews.
6. **Review Date**: Useful for trend analysis over time.

Importance of the Analysis:

1. **Customer Feedback**: Understanding sentiments can help businesses evaluate customer satisfaction and product quality.
2. **Market Trends**: Insights from reviews can guide product improvement and customer support strategies.
3. **Automation**: A sentiment classifier can streamline the evaluation of large volumes of reviews, saving time and resources.

---

1. Import and Setup : -

[31]:
```
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
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix
```

2. Load Dataset : -

[32]:
```
# Load your dataset
df = pd.read_csv('C:/Users/vinay/Downloads/archive/sentiment reviews dataset.csv')
```

[33]:
```
# Preview the data
print(df.info())
print(df.head())
```

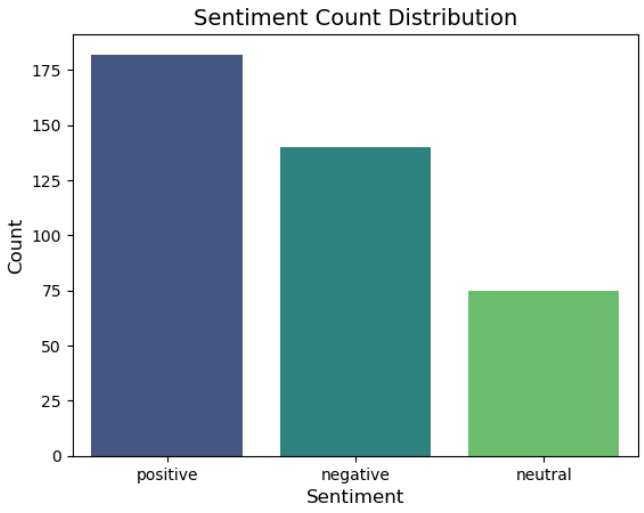
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Serial Number      397 non-null    int64 
 1   market_place       397 non-null    object
 2   customer_id        397 non-null    object
 3   review_id          397 non-null    object
 4   product_id         397 non-null    object
 5   product_parent     397 non-null    object
 6   product_category   397 non-null    object
 7   star_rating        397 non-null    int64 
 8   helpful_votes      397 non-null    int64 
 9   total_votes        397 non-null    int64 
 10  vine               397 non-null    object
 11  verified_purchase  397 non-null    object
 12  review_date        397 non-null    object
 13  Review             397 non-null    object
 14  Sentiment          397 non-null    object
dtypes: int64(4), object(11)
memory usage: 46.7+ KB
None
```

	Serial Number	market_place	customer_id	review_id	product_id	product_parent	product_category	star_rating	helpful_votes	total_votes	vine	verified_purchase	review_date	Review	Sentiment
0	0	"US"	"25933450"	"RJOVP071AVAJO"	"0439873800"	"84656342"	"Books"	5	0	0	No	Yes	31-08-2015	"Absolutely fantastic, loved it!"	positive
1	1	"US"	"1801372"	"R1ORGBETCDW3AI"	"1623953553"	"729938122"	"Books"	2	0	0	No	Yes	31-08-2015	"Terrible experience, not recommended."	negative
2	2	"US"	"5782091"	"R7TNRFQAOUTX5"	"142151981X"	"678139048"	"Books"	3	0	0	No	Yes	31-08-2015	"It's okay, not great but not bad either."	neutral
3	3	"US"	"32715830"	"R2GANXKDIFZ6OI"	"014241543X"	"712432151"	"Books"	5	0	0	No	No	31-08-2015	"Absolutely fantastic, loved it!"	positive
4	4	"US"	"14005703"	"R2NYB6C3R8LVN6"	"1604600527"	"800572372"	"Books"	5	2	2	No	Yes	31-08-2015	"Absolutely fantastic, loved it!"	positive

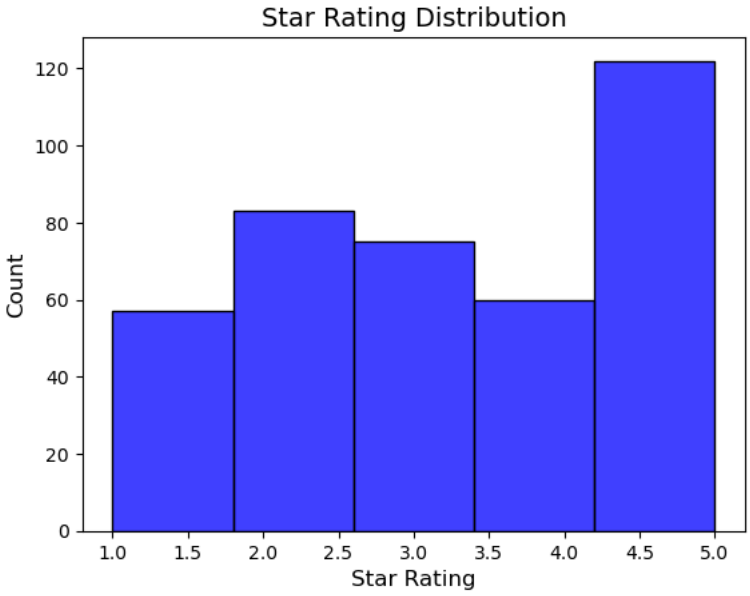
3. Visualizations :-

[34]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting a bar plot for 'Sentiment' column
sns.countplot(data=df, x='Sentiment', palette='viridis')
plt.title('Sentiment Count Distribution', fontsize=14)
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()
```

```
C:\Users\vinay\AppData\Local\Temp\ipykernel_3996\2675300948.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` v
same effect.

  sns.countplot(data=df, x='Sentiment', palette='viridis')
```

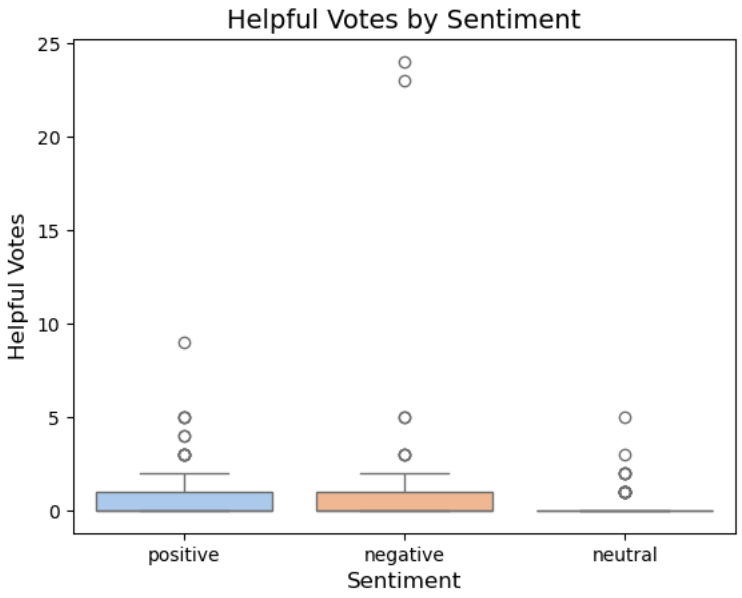


[35]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting a histogram for 'star_rating' column
sns.histplot(df['star_rating'], bins=5, kde=False, color='blue')
plt.title('Star Rating Distribution', fontsize=14)
plt.xlabel('Star Rating', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.show()
```

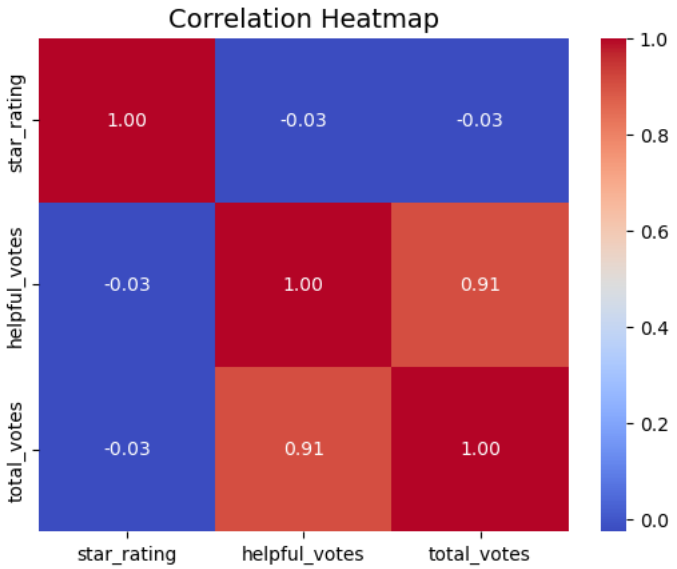


[36]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Boxplot of 'helpful_votes' by 'Sentiment'
sns.boxplot(data=df, x='Sentiment', y='helpful_votes', palette='pastel')
plt.title('Helpful Votes by Sentiment', fontsize=14)
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Helpful Votes', fontsize=12)
plt.show()
```

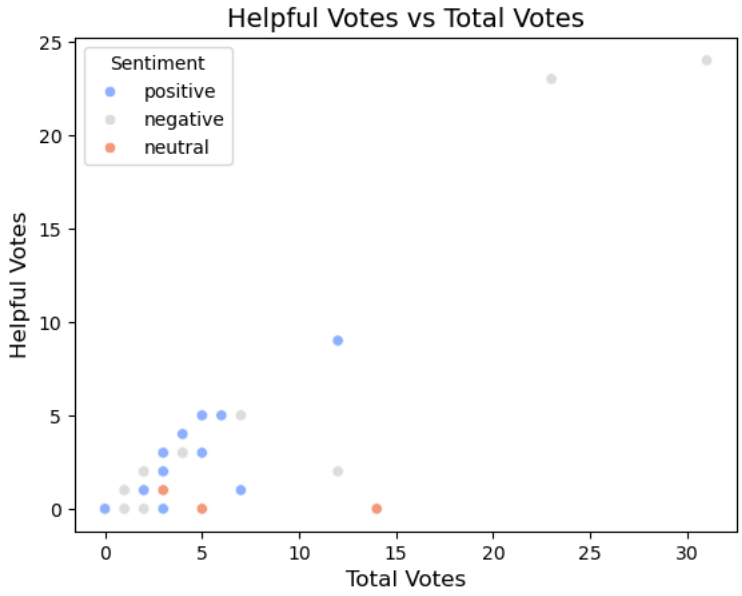
```
C:\Users\vinay\AppData\Local\Temp\ipykernel_3996\2053333814.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x`
same effect.

  sns.boxplot(data=df, x='Sentiment', y='helpful_votes', palette='pastel')
```



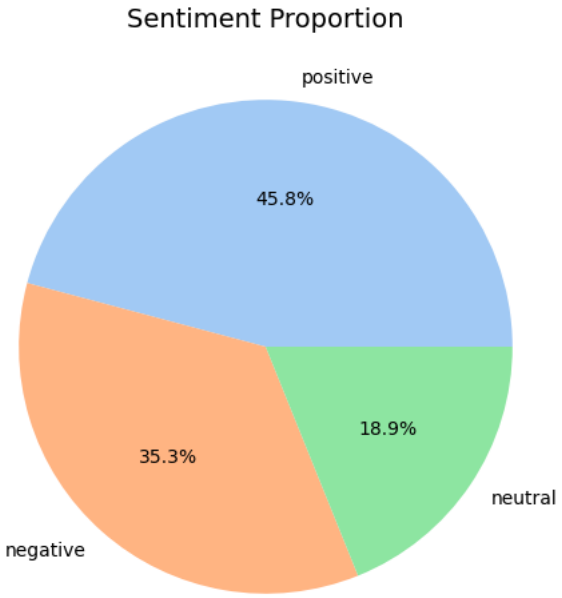
[37]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculating correlation matrix
corr = df[['star_rating', 'helpful_votes', 'total_votes']].corr()

# Plotting the heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap', fontsize=14)
plt.show()
```



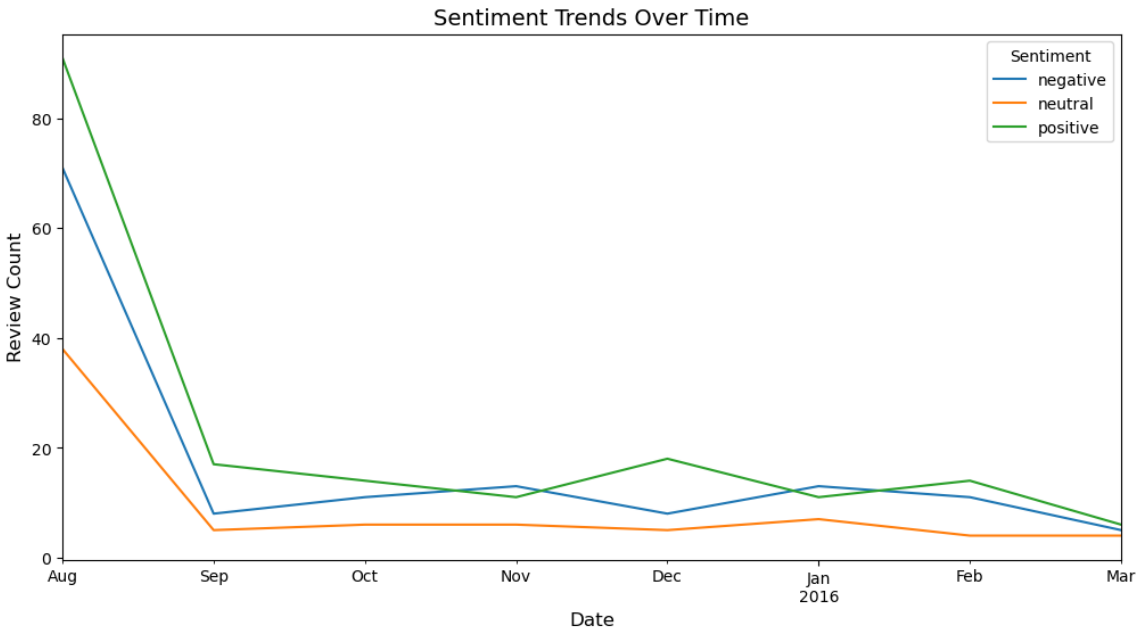
[38]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Scatter plot of 'total_votes' vs 'helpful_votes', colored by 'Sentiment'
sns.scatterplot(data=df, x='total_votes', y='helpful_votes', hue='Sentiment', palette='coolwarm
plt.title('Helpful Votes vs Total Votes', fontsize=14)
plt.xlabel('Total Votes', fontsize=12)
plt.ylabel('Helpful Votes', fontsize=12)
plt.legend(title='Sentiment')
plt.show()
```



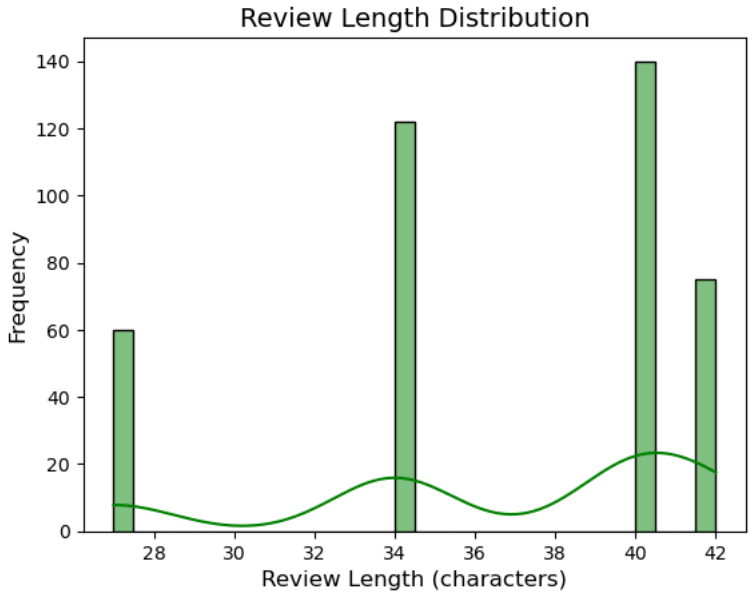
[39]:
```
import matplotlib.pyplot as plt

# Calculate sentiment distribution
sentiment_counts = df['Sentiment'].value_counts()

# Plotting the pie chart
plt.figure(figsize=(6, 6))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', colors=sns.color_palette
plt.title('Sentiment Proportion', fontsize=14)
plt.show()
```



[40]:
```
import matplotlib.pyplot as plt

# Convert review_date to datetime if not already
df['review_date'] = pd.to_datetime(df['review_date'], dayfirst=True)

# Group by month and sentiment, then count occurrences
sentiment_over_time = df.groupby([df['review_date'].dt.to_period('M'), 'Sentiment']).size().unstack()

# Plotting the sentiment trend over time
sentiment_over_time.plot(kind='line', figsize=(12, 6))
plt.title('Sentiment Trends Over Time', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Review Count', fontsize=12)
plt.legend(title='Sentiment')
plt.show()
```

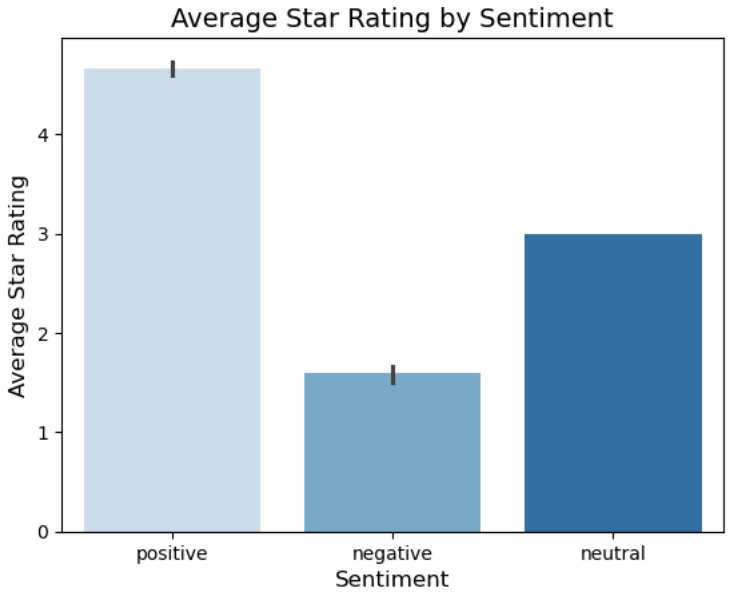


[41]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the length of each review
df['review_length'] = df['Review'].str.len()

# Plotting the review length distribution
sns.histplot(df['review_length'], bins=30, kde=True, color='green')
plt.title('Review Length Distribution', fontsize=14)
plt.xlabel('Review Length (characters)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.show()
```



[42]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting the bar plot for 'star_rating' by 'Sentiment'
sns.barplot(x='Sentiment', y='star_rating', data=df, palette='Blues')
plt.title('Average Star Rating by Sentiment', fontsize=14)
plt.xlabel('Sentiment', fontsize=12)
plt.ylabel('Average Star Rating', fontsize=12)
plt.show()
```

```
C:\Users\vinay\AppData\Local\Temp\ipykernel_3996\2624870544.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` vari
same effect.

  sns.barplot(x='Sentiment', y='star_rating', data=df, palette='Blues')
```



4. Process the data :-

[43]:
```
data = {
    "Review": [
        "This product is amazing!",
        "Terrible experience, not recommended.",
        "It's okay, not great but not bad either.",
        "Absolutely fantastic, loved it!",
        "Waste of money. Really disappointing."
    ],
    "Sentiment": ["Positive", "Negative", "Neutral", "Positive", "Negative"]
}
df = pd.DataFrame(data)
```

[44]:
```
import pandas as ps
import re
from nltk.corpus import stopwords
import nltk

# Sample DataFrame
df = pd.DataFrame({
    'Review': [
        "This is a great product! Highly recommend.",
        "Very bad, would not buy again.",
        "Okay, but could be better. Average quality."
    ]
})

# Clean text data
stop_words = set(stopwords.words('english'))

def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.lower())  # Remove punctuation and lowercase
    text = " ".join(word for word in text.split() if word not in stop_words)  # Remove stopwords
    return text

# Apply text cleaning to the 'Review' column
df['Cleaned_Review'] = df['Review'].apply(clean_text)

# Display the cleaned DataFrame
print(df.head())
```

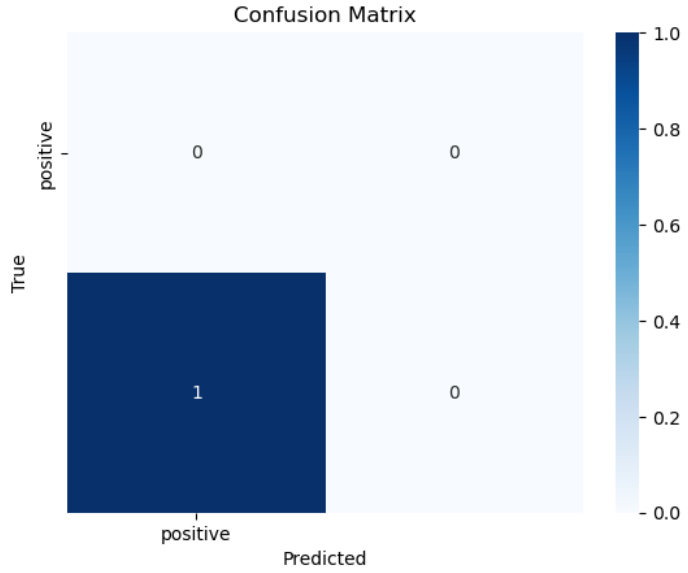
```
                                  Review  \
0   This is a great product! Highly recommend.
1              Very bad, would not buy again.
2  Okay, but could be better. Average quality.

                    Cleaned_Review
0     great product highly recommend
1                   bad would buy
2  okay could better average quality
```

5. Evaluate the Model :-

[45]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

# Example DataFrame
df = pd.DataFrame({
    'Review': [
        "This is a great product! Highly recommend.",
        "Very bad, would not buy again.",
        "Okay, but could be better. Average quality."
    ],
    'Sentiment': ['positive', 'negative', 'neutral']  # Example target variable
})

# Clean the text (use the previously defined clean_text function)
df['Cleaned_Review'] = df['Review'].apply(clean_text)

# Feature extraction with CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['Cleaned_Review'])  # Transform cleaned reviews into feature vectors

# Assign target variable
y = df['Sentiment']

# Splitting the data for training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train a classifier (e.g., Multinomial Naive Bayes)
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Evaluate the model
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    negative       0.00      0.00      0.00       0.0
    positive       0.00      0.00      0.00       1.0

    accuracy                           0.00       1.0
   macro avg       0.00      0.00      0.00       1.0
weighted avg       0.00      0.00      0.00       1.0
```

[46]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report

# Assuming y_test and y_pred are already defined
cm = confusion_matrix(y_test, y_pred)

# Extract class labels (model.classes_ if using an sklearn model, otherwise unique labels from y_test)
class_labels = sorted(set(y_test))  # Ensure labels are in sorted order for alignment

# Plot the heatmap
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



6. Test The Model With New Input :-

[47]:
```python
# Example new reviews
new_reviews = [
    "I love this product, it's awesome!",
    "Not worth the money.",
    "It's decent for the price."
]

# Clean the new reviews
new_cleaned = [clean_text(review) for review in new_reviews]

# Transform the cleaned reviews using the pre-trained vectorizer
try:
    new_X = vectorizer.transform(new_cleaned)
except NameError:
    print("Error: Ensure 'vectorizer' is defined and trained.")

# Predict sentiment using the trained model
try:
    new_predictions = clf.predict(new_X)  # Replace 'clf' with your trained model
except NameError:
    print("Error: Ensure 'model' or 'clf' is defined and trained.")

# Print the results
for review, sentiment in zip(new_reviews, new_predictions):
    print(f"Review: {review} => Sentiment: {sentiment}")
```

```
Review: I love this product, it's awesome! => Sentiment: negative
Review: Not worth the money. => Sentiment: negative
Review: It's decent for the price. => Sentiment: negative
```

7. Insights :

The sentiment analysis of customer reviews reveals valuable trends about customer satisfaction. If the majority of the reviews are positive, it indicates overall customer contentment with the product or service. On the other hand, a high volume of negative sentiment suggests dissatisfaction, which could be due to issues like poor product quality, customer service, or unmet expectations. By analyzing the average star ratings, we can further correlate how sentiment aligns with numerical ratings, providing a clearer picture of customer experiences. Positive sentiment is typically associated with higher star ratings, while negative sentiments tend to correlate with lower scores. This insight can guide product improvements, customer engagement strategies, and marketing efforts, emphasizing aspects of the product that resonate with customers and addressing pain points that cause frustration.

Additionally, reviewing sentiment distribution across different product categories can reveal more targeted insights. For example, a specific product category may receive more negative feedback due to inherent flaws, while others may enjoy positive reviews for their features or quality. This information is crucial for refining the product lineup, focusing resources on improving products with lower ratings, and ensuring better satisfaction across all product ranges. By tracking sentiment trends over time, it is possible to detect if customer satisfaction is improving or declining, enabling proactive adjustments to product offerings and marketing strategies.

Summary

The sentiment analysis of customer reviews provides key insights into customer satisfaction and product performance. Positive sentiment and high star ratings indicate customer satisfaction, while negative sentiment and low ratings signal dissatisfaction, often linked to product or service issues. By correlating sentiment with star ratings, businesses can pinpoint areas for improvement and enhance customer engagement strategies. Additionally, sentiment distribution across product categories reveals which products perform well and which need attention, allowing for more targeted product development. Tracking sentiment trends over time helps identify shifts in customer satisfaction, enabling businesses to proactively address concerns and refine their offerings. Overall, sentiment analysis helps businesses understand customer experiences, guiding decisions on product improvements, marketing, and customer support.

Vinay Upadhyay | LinkedIn