2-Exploratory Data Analysis

November 15, 2024

1 Exploratory Data Analysis (EDA)

In this section, we explore the filtered dataset to better understand trends and distributions. This analysis focuses on business ratings, review counts, and common keywords in customer reviews.

1.1 Goals

- Gain insights into the distribution of ratings and review counts.
- Identify frequently used keywords and trends in customer feedback.

1.2 Steps

1. **Statistical Overview**: Descriptive statistics for restaurants (e.g., average rating, review count).
2. **Visualization**: Use graphs to understand the distribution of ratings, popular restaurant types, and common keywords.
3. **Word Frequency Analysis**: Analyze the frequency of words used in reviews to identify the most discussed topics by customers.

[ ]:
```
import nltk
import string
import pandas as pd

# Download necessary resources
#nltk.download('punkt')
filtered_reviews = pd.read_csv('filtered_mid_range_business_reviews.csv')

# Function to clean and tokenize text
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    return tokens

# Apply text cleaning to the reviews
filtered_reviews['cleaned_text'] = filtered_reviews['text'].apply(clean_text)
```

```
# Display a sample of cleaned reviews
print("Sample of cleaned reviews:")
print(filtered_reviews[['text', 'cleaned_text']].head())
```

[ ]:
```
# Service-related keywords (covering positive, neutral, and negative aspects)
service_keywords = {
    'wait', 'waiting', 'waiter', 'waitress', 'slow', 'unfriendly', 'rude',␣
 ↪'impolite', 'service', 'staff', 'server',
    'host', 'hostess', 'inattentive', 'ignored', 'unhelpful', 'bad experience',␣
 ↪'customer service', 'attitude',
    'manners', 'hospitality', 'helpful', 'polite', 'courteous', 'friendly',␣
 ↪'accommodating', 'welcoming',
    'efficient', 'attentive', 'prompt', 'quick', 'delayed', 'long wait', 'poor␣
 ↪service', 'great service', 'excellent staff',
    'line', 'queue', 'customer care', 'bartender', 'manager', 'reservation',␣
 ↪'seated', 'seat', 'approachable',
    'service charge', 'unprofessional', 'courtesy', 'professional', 'dining␣
 ↪experience'
}

# Food-related keywords (broad coverage for flavor, preparation, etc.)
food_keywords = {
    'delicious', 'authentic', 'flavor', 'flavour', 'taste', 'fresh', 'spicy',␣
 ↪'cuisine', 'dish', 'quality', 'tasty',
    'savory', 'savoury', 'bland', 'undercooked', 'overcooked', 'portion',␣
 ↪'serving', 'appetizer', 'entree', 'dessert',
    'drink', 'beverage', 'spices', 'seasoning', 'hot', 'cold', 'crunchy',␣
 ↪'tender', 'sweet', 'salty', 'bitter', 'sour', 'umami',
    'texture', 'rich', 'balance', 'presentation', 'garnish', 'portion size',␣
 ↪'meal', 'menu', 'variety', 'specialty',
    'chef', 'authenticity', 'sauce', 'side dish', 'main course', 'buttery',␣
 ↪'crispy', 'greasy', 'baked', 'fried', 'grilled',
    'marinated', 'spicy', 'mild', 'flavorful', 'succulent', 'moist', 'juicy',␣
 ↪'dry', 'burnt', 'seasoned', 'prepared',
    'cooked', 'presentation', 'visual appeal', 'aroma', 'ingredients',␣
 ↪'organic', 'vegan', 'gluten-free', 'dietary restrictions'
}

# Ambiance-related keywords (covering restaurant atmosphere)
ambiance_keywords = {
    'ambiance', 'ambience', 'atmosphere', 'lighting', 'music', 'noise',␣
 ↪'decor', 'interior', 'seating', 'comfortable',
    'cozy', 'noisy', 'quiet', 'intimate', 'crowded', 'spacious', 'design',␣
 ↪'layout', 'setting', 'environment',
```

```
    'clean', 'vibe', 'energy', 'smell', 'aroma', 'air conditioning',␣
 ↪'ventilation', 'dim', 'bright', 'romantic',
    'rustic', 'modern', 'vintage', 'warm', 'welcoming', 'lively', 'pleasant',␣
 ↪'uncomfortable', 'drafty', 'temperature',
    'loud', 'peaceful', 'relaxed', 'elegant', 'luxurious', 'family-friendly',␣
 ↪'decorations', 'theme', 'aesthetic'
}

# Value-related keywords (covering cost, pricing, and value for money)
value_keywords = {
    'price', 'expensive', 'cheap', 'value', 'worth', 'affordable', 'deal',␣
 ↪'reasonable', 'cost', 'overpriced',
    'cheap', 'discount', 'offer', 'special', 'money', 'worth it', 'not worth␣
 ↪it', 'expensive for what you get',
    'underpriced', 'fair price', 'menu price', 'portion size vs price', 'value␣
 ↪for money', 'bargain', 'premium',
    'costly', 'high-end', 'budget', 'wallet-friendly', 'luxury', 'fees',␣
 ↪'service charge', 'hidden fees',
    'pay', 'bill', 'check', 'reasonable price', 'competitive pricing', 'fair␣
 ↪value'
}

# Cleanliness-related keywords (covering cleanliness and hygiene aspects)
cleanliness_keywords = {
    'clean', 'dirty', 'hygiene', 'sanitary', 'unsanitary', 'messy', 'tidy',␣
 ↪'smell', 'odor', 'sticky', 'bathroom',
    'restroom', 'toilet', 'floor', 'table', 'napkins', 'utensils', 'plates',␣
 ↪'glasses', 'mold', 'dust', 'bugs',
    'pests', 'rats', 'cockroaches', 'well-maintained', 'unkept', 'neat',␣
 ↪'spotless', 'filthy', 'stains', 'garbage',
    'trash', 'bins', 'soap', 'hand sanitizer', 'cleaning', 'wiped', 'cramped',␣
 ↪'cluttered', 'organized',
    'sanitizer', 'mop', 'vacuum', 'waste', 'bathroom cleanliness',␣
 ↪'disinfected', 'clean table', 'clean atmosphere'
}
```

[ ]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize VADER
sia = SentimentIntensityAnalyzer()

# Function to calculate sentiment for sentences with specific keywords
def keyword_sentiment_analysis(text, keywords):
    sentences = nltk.sent_tokenize(text)
    relevant_sentiments = []
    for sentence in sentences:
```

```
        if any(word in sentence for word in keywords):
            score = sia.polarity_scores(sentence)['compound']
            relevant_sentiments.append(score)
    if relevant_sentiments:
        return sum(relevant_sentiments) / len(relevant_sentiments)  # Average␣
 ↪score
    else:
        return None

# Apply sentiment analysis to relevant sentences for different aspects
filtered_reviews['service_sentiment'] = filtered_reviews['text'].apply(lambda x:
 ↪ keyword_sentiment_analysis(x, service_keywords))
filtered_reviews['food_sentiment'] = filtered_reviews['text'].apply(lambda x:␣
 ↪keyword_sentiment_analysis(x, food_keywords))
filtered_reviews['ambiance_sentiment'] = filtered_reviews['text'].apply(lambda␣
 ↪x: keyword_sentiment_analysis(x, ambiance_keywords))
filtered_reviews['value_sentiment'] = filtered_reviews['text'].apply(lambda x:␣
 ↪keyword_sentiment_analysis(x, value_keywords))
filtered_reviews['cleanliness_sentiment'] = filtered_reviews['text'].
 ↪apply(lambda x: keyword_sentiment_analysis(x, cleanliness_keywords))

# Display sample results
print(filtered_reviews[['service_sentiment', 'food_sentiment',␣
 ↪'ambiance_sentiment', 'value_sentiment', 'cleanliness_sentiment']].head())
```

[ ]:
```
# Save the new filtered reviews for future use
filtered_reviews.to_csv('filtered_exp_business_reviews.csv', index=False)
```