face_masks_analysis

March 17, 2025

1 Face Masks Exploratory Data Analysis and Visualization

1.1 Introduction

This notebook contains the analysis of face mask products and customer reviews data to provide insights to a manufacturer of personal care products. The objective is to understand the market competition, customer preferences, and segments to improve marketing strategy and R&D initiatives.

[ ]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from datetime import datetime
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.sentiment import SentimentIntensityAnalyzer
from wordcloud import WordCloud

# Set visualization style
plt.style.use('seaborn-whitegrid')
sns.set_palette('viridis')

# Configure plot settings
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 12

# Download necessary NLTK resources
try:
    nltk.data.find('tokenizers/punkt')
except LookupError:
    nltk.download('punkt')
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
```

```
    nltk.download('stopwords')
try:
    nltk.data.find('sentiment/vader_lexicon')
except LookupError:
    nltk.download('vader_lexicon')
```

1.2 Data Loading and Initial Exploration

[ ]:
```
# Load the data
products_df = pd.read_csv('products.tsv', sep='\t')
reviews_df = pd.read_csv('reviews.tsv', sep='\t')

# Display basic information about the datasets
print("Products dataset shape:", products_df.shape)
print("Reviews dataset shape:", reviews_df.shape)

# Show the first few rows of each dataset
print("\nProducts dataset head:")
products_df.head()
```

[ ]:
```
# Examine reviews dataset
print("\nReviews dataset head:")
reviews_df.head()
```

[ ]:
```
# Check for missing values in both datasets
print("Missing values in products dataset:")
print(products_df.isnull().sum())

print("\nMissing values in reviews dataset:")
print(reviews_df.isnull().sum())
```

1.3 Data Cleaning and Preparation

[ ]:
```
# Clean the product price column
products_df['product_price'] = pd.to_numeric(products_df['product_price'],␣
 ↪errors='coerce')

# Create a standardized rating column in reviews
# The ratingValue seems to be on a 10-50 scale (10=1 star, 20=2 stars, etc.)
reviews_df['rating_stars'] = reviews_df['ratingValue'] / 10

# Convert postedDate to datetime
reviews_df['postedDate'] = pd.to_datetime(reviews_df['postedDate'],␣
 ↪errors='coerce')

# Extract product type from product name
def extract_mask_type(name):
```

```
    name = name.lower()
    if 'kn95' in name or 'n95' in name or 'ffp2' in name or 'kf94' in name or␣
 ↪'respirator' in name:
        return 'Respirator'
    elif 'cotton' in name or 'reusable' in name or 'cloth' in name:
        return 'Cloth/Reusable'
    elif 'disposable' in name:
        return 'Disposable'
    elif 'copper' in name:
        return 'Copper'
    elif 'nano' in name:
        return 'Nano Technology'
    elif 'filter' in name:
        return 'Filter'
    else:
        return 'Other'

products_df['mask_type'] = products_df['product_name'].apply(extract_mask_type)

# Extract other product features
def extract_pack_size(name):
    # Try to find patterns like "10 Pack", "5 Count", etc.
    patterns = [r'(\d+)\s+Pack', r'(\d+)\s+Count', r'(\d+)\s+Masks?',␣
 ↪r'(\d+)\s+Mask']
    for pattern in patterns:
        match = re.search(pattern, name, re.IGNORECASE)
        if match:
            return int(match.group(1))
    # Default to 1 if no pack size found
    return 1

products_df['pack_size'] = products_df['product_name'].apply(extract_pack_size)

# Calculate price per mask
products_df['price_per_mask'] = products_df['product_price'] /␣
 ↪products_df['pack_size']

# Extract language information from reviews
reviews_df['review_language'] = reviews_df['languageCode'].apply(lambda x: x.
 ↪split('-')[0] if isinstance(x, str) else 'unknown')
```

1.4 Analysis of Product Features and Pricing

[ ]:
```
# Distribution of mask types
plt.figure(figsize=(10, 6))
mask_type_counts = products_df['mask_type'].value_counts()
sns.barplot(x=mask_type_counts.index, y=mask_type_counts.values)
```

```
plt.title('Distribution of Face Mask Types')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

[ ]:
```
# Analyze price distribution by mask type
plt.figure(figsize=(12, 6))
sns.boxplot(x='mask_type', y='price_per_mask', data=products_df)
plt.title('Price per Mask by Mask Type')
plt.ylabel('Price per Mask (AUD)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

[ ]:
```
# Calculate average ratings per product
product_ratings = reviews_df.groupby('productId')['rating_stars'].agg(['mean',␣
↪'count']).reset_index()
product_ratings.columns = ['product_id', 'avg_rating', 'review_count']

# Merge with product information
product_analysis = pd.merge(products_df, product_ratings, left_on='product_id',␣
↪right_on='product_id', how='left')

# Display the top-rated products
top_rated = product_analysis.sort_values(by=['avg_rating', 'review_count'],␣
↪ascending=False)
top_rated = top_rated[top_rated['review_count'] > 5]  # Filter for products␣
↪with more than 5 reviews
top_rated[['product_id', 'product_name', 'product_price', 'price_currency',␣
↪'mask_type', 'avg_rating', 'review_count']].head(10)
```

1.5 Customer Reviews Analysis

[ ]:
```
# Analyze ratings distribution
plt.figure(figsize=(10, 6))
sns.countplot(data=reviews_df, x='rating_stars')
plt.title('Distribution of Customer Ratings')
plt.xlabel('Rating (Stars)')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

[ ]:
```
# Function to clean and prepare text for analysis
def clean_text(text):
    if isinstance(text, str):
        # Convert to lowercase
```

```
        text = text.lower()
        # Remove special characters, numbers, etc.
        text = re.sub(r'[^\w\s]', '', text)
        # Tokenize
        tokens = word_tokenize(text)
        # Remove stopwords
        stop_words = set(stopwords.words('english'))
        tokens = [word for word in tokens if word not in stop_words]
        return ' '.join(tokens)
    return ''

# Apply text cleaning to review text
reviews_df['clean_review_text'] = reviews_df['reviewText'].apply(clean_text)

# For non-English reviews, use the translated text if available
mask = (reviews_df['review_language'] != 'en') & (~reviews_df['translation.
 ↪reviewText'].isna())
reviews_df.loc[mask, 'clean_review_text'] = reviews_df.loc[mask, 'translation.
 ↪reviewText'].apply(clean_text)
```

[ ]:
```
# Perform sentiment analysis
sia = SentimentIntensityAnalyzer()

# Function to get sentiment scores
def get_sentiment(text):
    if isinstance(text, str) and text.strip():
        return sia.polarity_scores(text)['compound']
    return 0

reviews_df['sentiment_score'] = reviews_df['clean_review_text'].
 ↪apply(get_sentiment)

# Group sentiment by mask type
# First, we need to merge reviews with products
reviews_with_product = pd.merge(reviews_df, products_df[['product_id',␣
 ↪'mask_type']],
                                left_on='productId', right_on='product_id',␣
 ↪how='left')

# Now analyze sentiment by mask type
sentiment_by_type = reviews_with_product.
 ↪groupby('mask_type')['sentiment_score'].agg(['mean', 'count']).reset_index()
sentiment_by_type.columns = ['mask_type', 'avg_sentiment', 'review_count']

# Plot average sentiment by mask type
plt.figure(figsize=(10, 6))
```

```
sns.barplot(x='mask_type', y='avg_sentiment', data=sentiment_by_type,␣
 ↪hue='review_count', dodge=False, palette='viridis')
plt.title('Average Sentiment by Mask Type')
plt.xlabel('Mask Type')
plt.ylabel('Average Sentiment Score (-1 to 1)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

[ ]:
```
# Create a word cloud of common terms in positive reviews
positive_reviews = reviews_df[reviews_df['sentiment_score'] > 0.
 ↪5]['clean_review_text']
positive_text = ' '.join(positive_reviews.fillna(''))

# Generate a word cloud for positive reviews
plt.figure(figsize=(12, 8))
wordcloud = WordCloud(width=800, height=400, background_color='white',␣
 ↪colormap='viridis',
                      max_words=100).generate(positive_text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Common Words in Positive Reviews')
plt.axis('off')
plt.tight_layout()
plt.show()
```

[ ]:
```
# Create a word cloud of common terms in negative reviews
negative_reviews = reviews_df[reviews_df['sentiment_score'] < -0.
 ↪3]['clean_review_text']
negative_text = ' '.join(negative_reviews.fillna(''))

# Generate a word cloud for negative reviews
plt.figure(figsize=(12, 8))
wordcloud = WordCloud(width=800, height=400, background_color='white',␣
 ↪colormap='magma',
                      max_words=100).generate(negative_text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Common Words in Negative Reviews')
plt.axis('off')
plt.tight_layout()
plt.show()
```

1.6 Consumer Segmentation Analysis

[ ]:
```
# Analyze review patterns by language/region
language_stats = reviews_df.groupby('review_language').agg({
    'rating_stars': 'mean',
    'sentiment_score': 'mean',
```

```
    'productId': 'count'
}).reset_index()
language_stats.columns = ['language', 'avg_rating', 'avg_sentiment',␣
 ↪'review_count']
language_stats = language_stats.sort_values('review_count', ascending=False)

# Plot average rating by language
plt.figure(figsize=(12, 6))
sns.barplot(x='language', y='avg_rating', data=language_stats.head(10),␣
 ↪hue='review_count', dodge=False, palette='viridis')
plt.title('Average Rating by Language/Region')
plt.ylabel('Average Rating (Stars)')
plt.xlabel('Language')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

[ ]:
```
# Identify different consumer segments based on their preferences
# First, extract key features from reviews
def has_keyword(text, keywords):
    if isinstance(text, str):
        text = text.lower()
        return any(keyword in text for keyword in keywords)
    return False

# Define different consumer segments based on review content
comfort_keywords = ['comfortable', 'soft', 'breathable', 'easy to breathe',␣
 ↪'breathe']
protection_keywords = ['protection', 'safe', 'secure', 'sealed', 'filter',␣
 ↪'filtering']
style_keywords = ['stylish', 'design', 'color', 'look', 'fashion', 'cute']
fit_keywords = ['fit', 'size', 'large', 'small', 'tight', 'loose']
price_keywords = ['price', 'expensive', 'cheap', 'worth', 'value']

# Create segment flags
reviews_df['comfort_focused'] = reviews_df['reviewText'].apply(lambda x:␣
 ↪has_keyword(x, comfort_keywords))
reviews_df['protection_focused'] = reviews_df['reviewText'].apply(lambda x:␣
 ↪has_keyword(x, protection_keywords))
reviews_df['style_focused'] = reviews_df['reviewText'].apply(lambda x:␣
 ↪has_keyword(x, style_keywords))
reviews_df['fit_focused'] = reviews_df['reviewText'].apply(lambda x:␣
 ↪has_keyword(x, fit_keywords))
reviews_df['price_focused'] = reviews_df['reviewText'].apply(lambda x:␣
 ↪has_keyword(x, price_keywords))
```

```
# Calculate the prevalence of each segment
segment_counts = {
    'Comfort Focused': reviews_df['comfort_focused'].sum(),
    'Protection Focused': reviews_df['protection_focused'].sum(),
    'Style Focused': reviews_df['style_focused'].sum(),
    'Fit Focused': reviews_df['fit_focused'].sum(),
    'Price Focused': reviews_df['price_focused'].sum()
}

# Plot segment distribution
plt.figure(figsize=(10, 6))
sns.barplot(x=list(segment_counts.keys()), y=list(segment_counts.values()))
plt.title('Distribution of Consumer Segments')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

[ ]:
```
# Analyze ratings by segment
segment_ratings = {
    'Comfort Focused':␣
 ↪reviews_df[reviews_df['comfort_focused']]['rating_stars'].mean(),
    'Protection Focused':␣
 ↪reviews_df[reviews_df['protection_focused']]['rating_stars'].mean(),
    'Style Focused': reviews_df[reviews_df['style_focused']]['rating_stars'].
 ↪mean(),
    'Fit Focused': reviews_df[reviews_df['fit_focused']]['rating_stars'].mean(),
    'Price Focused': reviews_df[reviews_df['price_focused']]['rating_stars'].
 ↪mean()
}

# Plot average ratings by segment
plt.figure(figsize=(10, 6))
sns.barplot(x=list(segment_ratings.keys()), y=list(segment_ratings.values()))
plt.title('Average Ratings by Consumer Segment')
plt.ylabel('Average Rating (Stars)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

1.7 Key Findings and Recommendations

Based on our analysis, we can draw the following insights and recommendations:

1. **Most Popular Mask Types**: The analysis reveals that [to be filled based on results]
2. **Key Features Valued by Consumers**:
    - [to be filled based on results]

- [to be filled based on results]

3. **Consumer Segments**: We identified several distinct consumer segments with different priorities:
   - [to be filled based on results]
   - [to be filled based on results]
4. **Marketing Strategy Recommendations**:
   - [to be filled based on results]
   - [to be filled based on results]
5. **R&D Recommendations**:
   - [to be filled based on results]
   - [to be filled based on results]

1.8 Additional Insights from External Data

To enhance our analysis, we could incorporate the following external data sources:

1. **COVID-19 Statistics**: Regional infection rates could help understand market demand patterns.
2. **Demographics**: Age, gender, and income data could provide insights into consumer segmentation.
3. **Seasonal Data**: Weather patterns may affect mask preferences (breathability in summer vs. warmth in winter).
4. **Competitive Analysis**: Price and feature data from other e-commerce platforms beyond iHerb.
5. **Social Media Sentiment**: Analysis of mask discussions on platforms like Twitter or Instagram.

1.9 Limitations and Future Improvements

Our analysis faces several limitations that could be addressed in future work:

1. **Limited Demographics**: We lack explicit user demographic information (age, gender, etc.).
2. **Selection Bias**: Reviews represent only a subset of customers who chose to leave feedback.
3. **Temporal Analysis**: A more detailed analysis of how preferences changed over time could be valuable.
4. **Feature Extraction**: More sophisticated NLP techniques could better extract features from reviews.
5. **Regional Specificity**: More granular regional analysis could reveal cultural preferences.