Technical Appendix

Data Visualization Design Project

Jan'23 Term

By: Group 6

Deeksha Chutani, 21F1002583
Param Chordiya, 21F1003953
Shagun Dwivedi, 21F1001731
Shaifali Vashishtha,21F1003257

Tools Used

- Excel, Google Sheets
- Python (along with matplotlib, nltk, numpy, pandas, plotly, seaborn, TextBlob, and WordCloud libraries)
- Flourish

Dataset Overview

The table below lists each of the files available for analysis with a short description of what is found in each one.

File Name	Description	Fields
products.tsv	This file contains a line for each mask product on the iHerb website.	product_id: id of the product product_name: name of the product product_price: price of the product price_currency: currency in which the price is mentioned product_availability: link that tells whether the product is available or not product_url: link to the product webpage source_url: link to the vendor website
reviews.tsv	This file contains a line for each review on any products in the products.tsv file.	abuseCount: number of times the review was reported as abusive customerNickname: name used by the customer

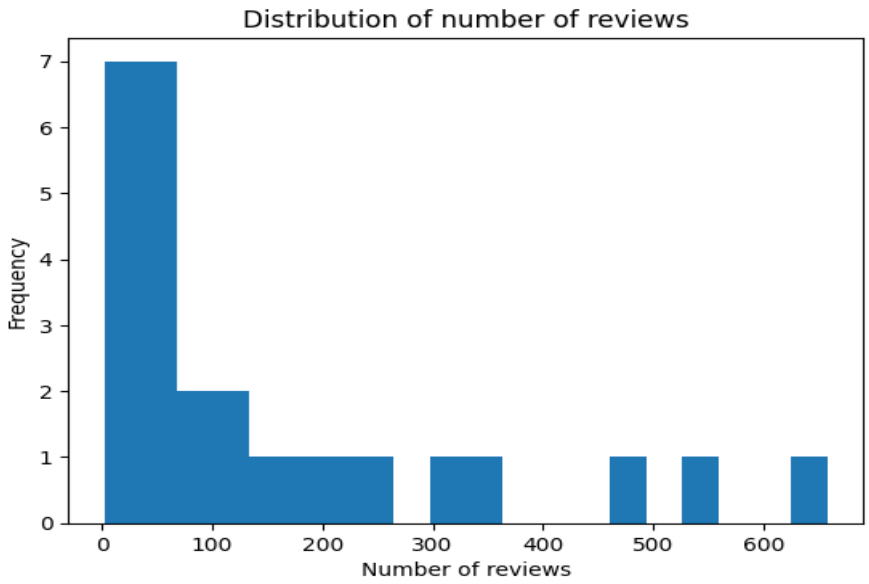
		helpfulNo: number of people who did not find the review helpful helpfulYes: number of people who found the review helpful id: id of the review imagesCount: number of images uploaded along with the review languageCode: language used in the review postedDate: date the review was posted productId: id for the product that was reviewed profileInfo.ugcSummary.answerCount: total number of answers provided by the user/reviewer profileInfo.ugcSummary.reviewCount: number of reviews by the user on the website ratingValue: rating given by the user for the product reviewText: original text of the review reviewTitle: original title of the review reviewed: whether the review has been reviewed by the vendor website score: score for the review translation.reviewText: review text translated into English translation.reviewTitle: review title translated into English

Aggregation

Total Number of Reviews: 4649
Number of Products: 27 items

Histogram showing distribution of reviews.

```
plt.hist(popularity_df['num_reviews'], bins=20)
plt.xlabel('Number of reviews')
plt.ylabel('Frequency')
plt.title('Distribution of number of reviews')
plt.show()
```

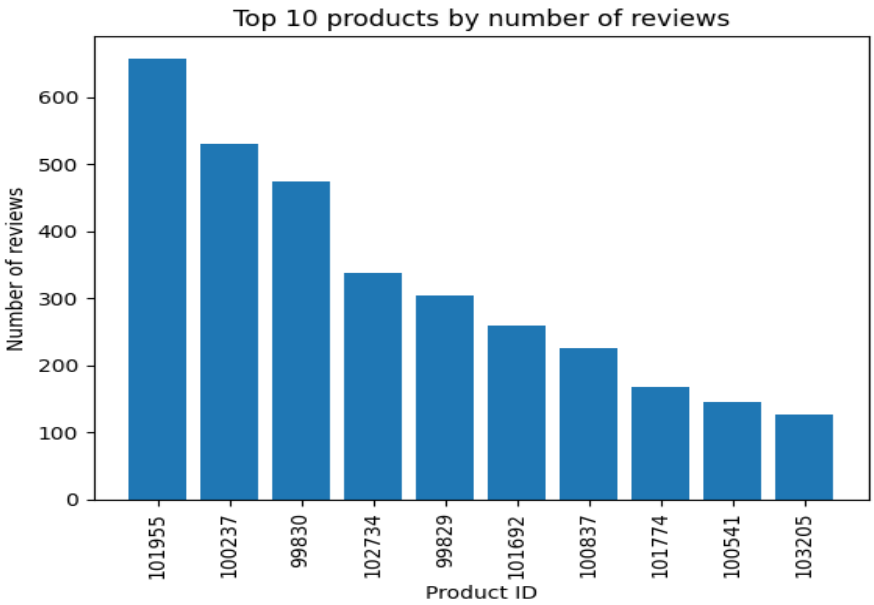


Filtering

Histogram showing number of reviews for top 10 products (ranked by number of reviews).

```
top10_df = popularity_df.nlargest(10, 'num_reviews')

plt.bar(top10_df['productId'], top10_df['num_reviews'])
plt.xticks(rotation=90)
plt.xlabel('Product ID')
plt.ylabel('Number of reviews')
plt.title('Top 10 products by number of reviews')
plt.show()
```



Data Preparation

For basic preparation, we removed any duplicate records, bringing down the number of records to 3765 in the reviews.tsv.

Attribute Creation

In reviews.tsv:

1) country: split languageCode into two attributes languageCode and country.

R	S
languageCode	country
en	US
ru	RU
en	US
ar	SA

2) newReviewText: created a column that contains only english review text, which is either original or translated.

V
newReviewText
good and comfy
Bought for the whole family for the period
It hurts the ears a little if you wear it for lo
Pretty like after washing
4
Recommend!

3) polarity and **Sentiment:** carried out sentiment analysis on the newReviewText using Textblob, the python library.

```
rev_df['polarity'] = rev_df['newReviewText'].apply(lambda x: TextBlob(str(x)).sentiment.polarity)
rev_df['Sentiment'] = rev_df['polarity'].apply(lambda x: 'negative' if x<0 else 'positive')
rev_df.head()
```

ıntry	translation.reviewText	translation.reviewTitle	newReviewText	polarity	Sentiment
US	NaN	NaN	good and comfy	0.70000	positive
RU	Bought for the whole family for the period of ...	en-US	Bought for the whole family for the period of ...	0.36000	positive

In products.tsv:

1) product_model, product_features: split product_name into product_name, product_model, and product_features.

	product_name	product_model	product_features
)5	Hwipure	Disposable KF94 (N95 / KN95/ FFP2) Mask	1 Mask,,
'4	HIGUARD	Disposable KF94 (N95 / KN95/ FFP2) Mask	1 Mask,,
)5	SunJoy	KN95 Professional Protective Disposable Face Ma	10 Pack,,
)8	Lozperi	Copper Mask	Adult,Black,1 Mask

2) Size, PackOf, Color: split product_features into Size, PackOf, and Color attributes.

3) Price/Mask: derived price of one mask by dividing the product_price by the PackOf column.

	Size	PackOf	Color	Country_Code	Price/Mask	Co
N95 / K	Free Size	20	Neutral	US	2.362	Un
N95 / K	Free Size	20	Neutral	US	2.362	Un
tive Ma	Free Size	50	Neutral	RU	0.307	Ru
tive Ma	Free Size	50	Neutral	US	0.307	Un

Merging the Datasets

This is the dataset that's primarily used and modified further in the analysis.

```
df = pd.merge(rev_df,prod_df, left_on=['productId'], right_on=['product_id'])
df.head()
```

	imagesCount	languageCode	postedDate	productId	profileInfo.ugcSummary.answerCount	...	product_model	product_features
	0	en-US	2021-02-06T12:46:05.712Z	99829	0.0	...	Nano Reusable Face Protection Mask	Large,1 Mask,
	0	ru-RU	2021-02-06T07:01:31.423Z	99829	0.0	...	Nano Reusable Face Protection Mask	Large,1 Mask,

Creation of the Visualisations

Creating Bar Chart for Popularity

count_df contains a count of reviews for each product along with some other features.

```
count_df = pd.DataFrame(df.product_id.value_counts())
add_brand = list()
add_model = list()
add_features = list()
for i in count_df.index:
    add_brand.append(prod_df[prod_df['product_id'] == i].product_name.iloc[0])
    add_model.append(prod_df[prod_df['product_id'] == i].product_model.iloc[0])
    add_features.append(prod_df[prod_df['product_id'] == i].product_features.iloc[0].rstrip(', ,'))
count_df['brand'] = pd.Series(add_brand).values
count_df['model'] = pd.Series(add_model).values
count_df['features'] = pd.Series(add_features).values
count_df = count_df.rename(columns={'product_id':'count'})
count_df.head()
```

	count	brand	model	features
101955	658	SunJoy	KN95 Professional Protective Disposable Face Mask	10 Pack
100237	530	Kitsch	100% Cotton Reuseable Face Masks	Leopard,3 Pack
99830	475	Kosette	Nano Reusable Face Protection Mask	Medium,1 Mask
102734	337	Zidian	Disposable Protective Mask	50 Pack

We created an interactive bar chart using plotly, and made some customisations to suit the presentation style:

```
fig = px.bar(
    count_df.sort_values(by='count',ascending=False),x='brand',y='count',
    color='model', hover_name="model", hover_data=["features"],text_auto=True,
    color_discrete_sequence=px.colors.qualitative.Set2,
    labels=dict(brand="Brand Name", count="No. of Reviews"))

fig.update_layout(
    plot_bgcolor='rgba(245,242,238,100)',
    showlegend=False, paper_bgcolor='rgba(245,242,238,100)'
)
fig.write_html('top5products.html')
fig.show()
```

Creating Bar Chart for Popularity

sentiment_df contains a count of positive and negative reviews for each brand.

```
sentiment_df = pd.DataFrame(df[['product_name','Sentiment']].value_counts()).reset_index().sort_values(
    by='product_name').rename(columns={0:'count'}).sort_values(by='Sentiment',ascending=False)
sentiment_df.head()
```

product_name	Sentiment	count
Luseta Beauty	positive	72
Landsberg	positive	38
YJ Corporation	positive	61
Tony Moly	positive	133

Creating stacked bar chart for analysing the percentage of reviews by sentiment:

```
fig = px.histogram(sentiment_df, x="product_name", y="count",
             color='Sentiment', barnorm='percent',text_auto='.2f',
             color_discrete_sequence=px.colors.qualitative.Set2,
             labels=dict(product_name="Brand", count="Review Counts"))

fig.update_layout(
    plot_bgcolor='rgba(245,242,238,100)',
    paper_bgcolor='rgba(245,242,238,100)'
)
fig.write_html('sentiment_analysis_100stack.html')
fig.show()
```

Creating Word Clouds for Each Sentiment

We used the following libraries to analyse the reviews and create word clouds.

```
import pandas as pd
import numpy as np
import re #used as a regular expression to find particular patterns and process it
import string

from wordcloud import WordCloud
import seaborn as sns
import matplotlib.pyplot as plt

import nltk #a natural language processing toolkit module associated in anaconda
from nltk.corpus import stopwords
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer,SnowballStemmer
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
# # uncomment these for the first run
# nltk.download('punkt')
# nltk.download('averaged_perceptron_tagger')
# nltk.download('stopwords')
# nltk.download('omw-1.4')
# nltk.download('wordnet')
```

Processing, and simplifying the text for analysis:

```
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt


df['newReviewText'] = np.vectorize(remove_pattern)(df['newReviewText'].astype(str), "@[\w]*")
```

— Please Scroll Further —

```
# Define function for removing special characters
def remove_special_characters(text, remove_digits=True):
    pattern=r'[^a-zA-z0-9\s]'
    text=re.sub(pattern,'',text)
    return text

# Text-encoding: UTF-8 encoder
def to_unicode(text):
    if isinstance(text, float):
        text = str(text)
    if isinstance(text, int):
        text = str(text)
    if not isinstance(text, str):
        text = text.decode('utf-8', 'ignore')
    return text
# Removing emojis
def deEmojify(text):
    regrex_pattern = re.compile(pattern = "["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags = re.UNICODE)
    return regrex_pattern.sub(r'',text)
```

```
# Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

# Removing the noisy text
def denoise_text(text):
    text = to_unicode(text)
    text = re.sub(r"http\S+", "", text)
    text = deEmojify(text)
    text = text.encode('ascii', 'ignore')
    text = to_unicode(text)
    text = remove_between_square_brackets(text)
    text = remove_special_characters(text)
    text = text.lower() # lower case
    return text
```

Tokenising the text:

```
#Tokenization of text
tokenizer=ToktokTokenizer() #for every function

#Setting English stopwords
stopword_list=nltk.corpus.stopwords.words('english')
stopword_list.extend(['kf94','n95','don39t','mask','it39s','couldn39t','can39t','didn39t','good','excellent','great'])
```

```
# can remove stopwords either before or after stemming. But since this is a review context,
# we expect users to have used many different words and we did
# stemming before filtering for stopwords.

stop=set(stopwords.words('english'))
print(stop)

#Removing the stopwords
def remove_stopwords(text, is_lower_case=False):
    tokens = tokenizer.tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stopword_list]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stopword_list]
        filtered_text = ' '.join(filtered_tokens)
    return filtered_text

#Apply function on review column
df['newReviewText']=df['newReviewText'].apply(remove_stopwords)
```

```
def simple_stemmer(text):
    ps = SnowballStemmer(language='english')
    return ' '.join([ps.stem(word) for word in tokenizer.tokenize(text)])

def lemmatize_all(sentence):
    wnl = WordNetLemmatizer()
    for word, tag in pos_tag(word_tokenize(sentence)):
        if tag.startswith("NN"):
            yield wnl.lemmatize(word, pos='n')
        elif tag.startswith('VB'):
            yield wnl.lemmatize(word, pos='v')
        elif tag.startswith('JJ'):
            yield wnl.lemmatize(word, pos='a')
        else:
            yield word

def lemmatize_text(text):
    return ' '.join(lemmatize_all(text))
```

Creating separate lists for positive and negative words:

```
pos_list = " ".join([sentence for sentence in df[df['Sentiment']=='positive'].newReviewText]).replace('mask','')
neg_list = " ".join([sentence for sentence in df[df['Sentiment']=='negative'].newReviewText]).replace('mask','')
```

Finally, creating Word Clouds:

```
wordcloud = WordCloud(
    width=800, height=500, random_state=42, max_font_size=100,
    background_color="#f5f2eeff",colormap='tab20b',collocations=False).generate(pos_list)

# plot the graph
plt.figure(figsize=(8,8),facecolor='#f5f2eeff')
plt.title('Positive Reviews')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Creating HeatMap for Ratings of Each Brand

Creating a Pivot Table:

```
matrix = df.pivot_table(index='Product name', columns='ratingValue', values='customerNickname', aggfunc='count')
matrix.head()
```

ratingValue	10	20	30	40	50
Product name					
Dr. Puri	1.0	2.0	2.0	9.0	26.0
HIGUARD	2.0	3.0	11.0	31.0	121.0

Creating the HeatMap:

```
sns.set(font_scale=1.2)
sns.set_style('white')
fig, ax = plt.subplots(figsize=(20, 50))
sns.heatmap(matrix, annot=True, cmap='coolwarm', square=True, cbar=True)

plt.xlabel('Rating Value')
plt.ylabel('Product Name')
plt.title('Customer Ratings by Product')

plt.savefig('heatmap300.png',dpi=300)
plt.show()
```